TRABAJO FIN DE MÁSTER

MÁSTER UNIVERSITARIO OFICIAL EN CIENCIA DE DATOS E INGENIERÍA DE COMPUTADORES

Deep Learning para la simulación de sistemas físicos

Autor
Félix Fernández de la Mata

Directores
Miguel Molina Solana
Juan Gómez Romero

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, 14 de septiembre de 2021

Deep Learning para la simulación de sistemas físicos

Autor
Félix Fernández de la Mata

Directores
Miguel Molina Solana
Juan Gómez Romero

Deep Learning para la simulación de sistemas físicos

Félix Fernández de la Mata

Palabras clave: Deep Learning, Redes Neuronales Profundas, Physics Informed, Modelaje de Fenómenos Físicos, SciANN, PINN, Método de los Elementos Finitos, Ecuaciones Diferenciales en Derivadas Parciales.

Resumen

En la última década ha aumentado el número y la variedad de técnicas disponibles para construir y ajustar una red neuronal artificial. Una de estas nuevas técnicas son las *Physics-Informed Neural Networks* (PINNs). Este nuevo método permite la simulación de fenómenos físicos combinando tanto datos obtenidos de sensores como conocimiento de fenómenos físicos escrito en forma de ecuaciones diferenciales.

El modelaje de sistemas físicos es una técnica muy extendida que se emplea para el análisis y predicción del comportamiento de fenómenos físicos comunes como el sonido, el calor, el magnetismo o los fluidos. Aunque existen otras técnicas, los modelos PINN son más fáciles de construir y no necesitan tanta información del fenómeno simulado para obtener una predicción. Sin embargo, estas redes tienen actualmente un peor rendimiento que los métodos numéricos modernos, ya que tienden a dar peores predicciones y necesitan mucho más tiempo de computación para ajustarse.

Los objetivos del trabajo que aquí se presenta son múltiples. En primer lugar, se pretende ofrecer una explicación concisa de dos de los métodos clásicos de simulación de fenómenos físicos más conocidos, junto con una explicación suficientemente detallada de los mecanismos que permiten a los PINN construir dichas simulaciones a partir de datos escasos y ecuaciones diferenciales en derivadas parciales. En segundo lugar, se pretende evaluar la precisión de las predicciones que las PINNs obtienen con diferentes parámetros de entrenamiento. Para ello, se propondrán diferentes ejemplos teóricos y se compararán las predicciones de las PINN y de los métodos clásicos. Por último, se pretende discernir qué vías de desarrollo deben seguirse para mejorar el estado actual de la técnica PINN.

Deep Learning for simulation of physical systems

Félix Fernández de la Mata

Keywords: Deep Learning, Deep Neural Networks, Physics Informed, Physics Modelling, SciANN, PINN, Finite Element Method, Partial Differential Equations.

Abstract

Last decade has seen a rise in the number and variety of techniques available for building and fitting an artificial neural network. One of these new techniques are *Physics-Informed Neural Networks* (PINNs). This new method allows the simulation of physical phenomena by combining both data obtained from sensors and physics knowledge written in the form of differential equations.

Modeling of physical systems is a widespread technique used for analysis and prediction of the behavior of common physical phenomena as sound, heat, magnetism, or fluids. Although other techniques are available, PINNs are easier to build, and don't need as much knowledge of the simulated phenomenom in order to obtain a prediction. However, these networks currently have a worse performance than modern numerical methods, because they tend to yield worse predictions and need far more computing time to fit themselves.

The goals of the work presented here are multiple. Firstly, it aims to provide a concise explanation of two of the most widely known classical methods of physics simulation, along with a sufficiently detailed explanation of the mechanisms that allow PINNs to construct such simulations based on scarce data and partial differential equations. Secondly, it aims to evaluate the accuracy of the predictions PINNs obtain with different training parameters. This will be done by proposing different theoretical examples, and comparing the predictions of both PINNs and classical methods. Lastly, it aims to discern which development pathways should be followed in order to improve the current state of the PINN technique.

Yo, **Félix Fernández de la Mata**, alumno de la titulación Máster en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI **70911075M**, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Máster en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Félix Fernández de la Mata

Granada a 14 de septiembre de 2021.

D. **Miguel Molina Solana**, Profesor del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

D. **Juan Gómez Romero**, Profesor del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Deep Learning para la simulación de sistemas físicos***, ha sido realizado bajo su supervisión por **Félix Fernández de la Mata**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 14 de septiembre de 2021.

Los directores:

Miguel Molina Solana **Juan Gómez Romero**

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción.

El presente documento se corresponde con la memoria del Trabajo de Fin de Máster del alumno Félix Fernández de la Mata para el Máster Universitario en Ciencia de datos e Ingeniería de computadores de la Universidad de Granada (UGR) en el curso 2020/21. El tiempo de realización del conjunto de actividades relacionadas con la escritura de dicha memoria (revisión bibliográfica, planteamiento teórico, experimentación, etc.) está estimado en alrededor de 300 horas que corresponden a los 12 créditos ECTS asignados a esta actividad.

1.1. Preámbulo.

A menudo, el proceso de diseño de un producto ha de atravesar en varias ocasiones una fase de análisis de diseño mediante modelado por ordenador. El objetivo de esta fase es localizar problemas mecánicos o de planificación en el producto antes de comenzar a construir prototipos, con el objetivo de reducir gastos en dicha construcción. La tarea de construir estos modelos resulta bastante compleja, pues se basa en la construcción de modelos numéricos mediante software comercial, operado por personal especializado en este.

Durante los últimos años, el auge del Deep Learning y la proliferación de técnicas de entrenamiento ha hecho posible plantear nuevos métodos de modelaje que permiten analizar estas características físicas. Aunque estos planteamientos basados en Deep Learning aún no alcanzan la precisión de los modelos numéricos, tienen una implementación mucho más simple que los modelos numéricos, y permiten incorporar datos procedentes de experimentación real más fácilmente. Más concretamente, existe un tipo de red neuronal conocida como

PINN (*Physics-Informed Neural Network* [15]) que permite combinar todo el conocimiento relativo a la física que utilizan los modelos numéricos, y los datos obtenidos mediante experimentación para la construcción de modelos. Estas PINNs son bastante versátiles, pues son capaces de suplir la falta de datos reales mediante el uso de ecuaciones diferenciales, e inversamente, suplir el desconocimiento de propiedades físicas a través de datos.

Sería por tanto interesante conocer en qué medida permite el estado actual de las PINNs generar modelos que consigan los resultados de los modelos numéricos. En primer lugar, es necesario conocer hasta qué punto pueden aproximar la precisión de un modelo numérico, y qué parámetros consiguen predicciones más fiables. También puede ser interesante comprobar cómo evoluciona esta precisión máxima en función de la complejidad del problema propuesto. Por último, es importante estudiar el tiempo de cálculo necesario para alcanzar dicha precisión máxima para poder determinar si este nuevo método es viable como alternativa a los métodos numéricos.

1.2. Objetivos y tareas del trabajo.

El objetivo de este trabajo es doble. En primer lugar se realizará una presentación formal de los métodos de modelaje de fenómenos físicos que existen actualmente, comentando tanto su base teórica como sus métodos de implementación, y se explicarán sus puntos fuertes y desventajas. En segundo lugar, se profundizará en el uso de PINNs para la generación de esta clase de modelos a través de varios experimentos de complejidad creciente, para así poder evaluar cuantitativamente la utilidad de estas en su estado actual. Para realizar esta evaluación se han propuesto distintos experimentos (problemas de dificultad creciente) los cuales se intentarán resolver utilizando una PINN variando entre distintos parámetros. Para realizar esta evaluación de las PINNs, se han generado modelos gemelos mediante otras técnicas, cuyos resultados han sido utilizados como datos de validación. Todo el software necesario para la experimentación es de uso libre, y se encuentra disponible en los enlaces correspondientes de la bibliografía.

Por tanto, ha sido necesario realizar las siguientes tareas:

- Investigación sobre la base teórica de las tres técnicas presentadas en la memoria: planteamiento y resolución de ecuaciones diferenciales simbólicamente, modelos numéricos y uso de redes neuronales.
- Aprendizaje sobre los métodos de implementación de las tres técnicas

mencionadas.

- Definición de un conjunto de experimentos de dificultad creciente.
- Construcción de modelos simbólicos y numéricos para obtención de datos de validación.
- Construcción de modelos basados en PINN experimentación sobre una rejilla de parámetros.
- Análisis de los resultados de la experimentación.

El código generado durante el transcurso del proyecto se encuentra disponible en el repositorio [3].

La memoria está estructurada del siguiente modo: en el Capítulo 2 se hace una breve introducción teórica a la metodología de construcción de modelos físicos mediante el uso de ecuaciones diferenciales. Aquí se explica qué aporta una ecuación diferencial a la comprensión del comportamiento de un fenómeno físico, y qué técnicas se pueden utilizar para aprovechar este conocimiento a la hora de realizar el modelado. En el Capítulo 3 se realiza una introducción al apartado práctico del modelaje de estos fenómenos utilizando los métodos anteriormente explicados. Además, se da una explicación del entorno experimental que se ha utilizado, centrándose en la descripción de los experimentos propuestos y también en las métricas utilizadas para evaluar la precisión de una PINN respecto de los datos de validación. En el Capítulo 4 se realiza un análisis de los resultados de la experimentación, comentando cada experimento de manera separada y dando una breve explicación de comportamientos interesantes o anómalos que se han observado. Por último, en el Capítulo 5 se discuten las conclusiones extraídas de la experimentación atendiendo a las ventajas y desventajas de las PINNs respecto de otras técnicas, y se plantean varias vías de mejora y desarrollo para las PINNs.

1.3. Planificación.

Para la realización del proyecto se ha realizado una estimación temporal en semanas de la duración de cada tarea. En la Tabla 1.1 se muestra una relación de estas tareas junto con su duración y sus fechas estimadas de inicio y cierre.

Las tareas T1.1 y T1.2 se han realizado simultáneamente, y consisten en la familiarización con la base teórica y las librerías relativas al código del proyecto (SciANN y MLflow).

La tarea T2 consiste en la construcción de un cuaderno modelo que se ha utilizado como plantilla en la experimentación y que permite realizar fácilmente el proceso de experimentación. La tarea T3 consiste en la definición de los problemas que componen cada experimento en forma de sistema de ecuaciones diferenciales. En la tarea T4 se han obtenido los datos de validación correspondientes a cada experimento a través de métodos numéricos o mediante la resolución del sistema de ecuaciones.

La tarea T5 y sus subtareas componen la parte de experimentación del proyecto. Cada una de las tres tareas consiste en la construcción de un modelo PINN, experimentación sobre una rejilla de parámetros, y evaluación de resultados.

La tarea T6 es la compilación de toda la base teórica adquirida, la descripción del entorno de experimentación, y el análisis de resultados.

ID	**Tarea**	**Duración (Semanas)**	**Inicio**	**Fin**
T1.1	Revisión bibliográfica	3	31/05/21	18/06/21
T1.2	Pruebas iniciales	3	31/05/21	18/06/21
T2	Construcción del entorno experimental	1	21/06/21	25/06/21
T3	Definición de experimentos	1	28/06/21	2/07/21
T4	Obtención de datos de validación	1	5/07/21	9/07/21
T5	Experimentación	8	12/07/21	3/09/21
T5.1	Experimento 1	1	12/07/21	16/07/21
T5.2	Experimento 2	2	19/07/21	30/07/21
T5.3	Experimento 3	2	2/08/21	13/08/21
T5.4	Experimento 4	1	16/08/21	3/09/21
T6	Redacción de la memoria	5	2/08/21	3/09/21

Cuadro 1.1: Tareas del proyecto y su duración.

Capítulo 2

Definiciones y estado del arte.

En este capítulo se explicará brevemente el concepto de descripción de un fenómeno físico mediante ecuaciones diferenciales, y se darán algunos ejemplos básicos para ilustrarlo. Después se hablará sobre el método clásico para resolver estas ecuaciones diferenciales de manera simbólica. A continuación se verá cómo funciona la resolución mediante métodos numéricos y sus ventajas y desventajas respecto del método clásico.

Por último, se incluirá una explicación más extensa sobre la aplicación de redes neuronales a la simulación de estos fenómenos. En concreto, se explicará cómo funcionan las Physics-Informed Neural Networks (PINN) y se introducirá la biblioteca de Python que se ha usado para llevar a cabo la experimentación en esta memoria.

2.1. Descripción de un fenómeno físico mediante ecuaciones diferenciales.

A menudo resulta necesario describir un fenómeno físico concreto. Ejemplos de esto pueden ser el problema de lanzar un proyectil a una distancia específica, querer conocer la evolución de la temperatura en un edificio o las turbulencias del aire en unas condiciones dadas. Por desgracia, la tarea de obtener una función (o un conjunto de ellas) que describan tal sistema no suele ser trivial, pues el estado del sistema en un momento t y un punto p concretos depende del estado en el momento anterior de los puntos cercanos a p, este incluido. Por esta razón suele resultar más conveniente describir las interacciones de todos los puntos (o elementos) del sistema mediante una ecuación diferencial, para después derivar

una solución simbólica, si es posible, y en caso contrario, construir una solución mediante métodos numéricos.

Para ilustrar el proceso de la obtención de una ecuación diferencial que describe un fenómeno físico, vamos a explicar cómo se deriva la ecuación clásica del calor siguiendo la explicación mostrada en [5]:

La idea de la que partimos es la siguiente: La variación del calor en una región es igual a la pérdida de calor a través del borde. Vamos a representarla simbólicamente.

Supongamos que tenemos una región cerrada del espacio $\Omega \subset R^3$ con borde $\partial\Omega$. Denotaremos como q la función que describe el calor en cada punto de Ω y h representará el vector flujo de calor en el borde de dicha superficie. La ecuación que describe el enunciado anterior es:

$$\frac{d}{dt}\int_{\Omega} q\,dx = -\int_{\partial\Omega} h \cdot n\,dS.$$

Donde:

- $\frac{d}{dt}$ representa la variación a lo largo del término de la integral sobre la que se aplica.
- $\int_{\Omega} q\,dx$ representa el calor total a lo largo de todo Ω.
- $-\int_{\partial\Omega} h \cdot n\,dS$ representa la pérdida de calor (de ahí el signo negativo), a lo largo del borde de Ω. El integrando $h \cdot n$ representa la cantidad de calor que atraviesa dicho borde en dirección perpendicular a este en cada punto.

Ahora, aplicando el teorema de Stokes a la integral que describe la pérdida de calor, y utilizando el teorema fundamental del cálculo en el primer miembro, obtenemos

$$\int_{\Omega} q_t\,dx = -\int_{\Omega} \nabla h \cdot n\,dS.$$

Como esta ecuación se cumple para cualquier región Ω, podemos escribirla de la siguiente manera:

$$q_t = -\nabla h. \tag{2.1}$$

Ahora vamos a aplicar dos consideraciones que se cumplen en la mayoría de casos (una excepción a esta regla son, por ejemplo, los materiales muy porosos). En primer lugar, consideraremos que el calor u es proporcional a la temperatura según la ecuación $h = \rho c u$, donde ρ es la densidad del material que compone Ω, y c su capacidad calorífica. En segundo lugar, aplicaremos la ley de Fourier, que dice que el flujo de calor h es proporcional a la variación de la temperatura según la ecuación $h = -k\nabla u$, donde k es la conductividad térmica del material. Sustituyendo en (Ecuación 2.1) nos queda:

$$(\rho c u)_t = -\nabla\left(-k\nabla u\right)$$

Aplicando ahora que ρ y c son constantes en el tiempo, que k lo es en el espacio, y que como operadores $\nabla \circ \nabla \equiv \Delta$, resulta:

$$\rho c u_t = k\Delta u$$

Por último, asumiendo por simplicidad que $\rho = c = k = 1$, resulta

$$u_t = \Delta u \tag{2.2}$$

Que es la formulación más general de la ecuación del calor, y que no tiene en cuenta fuentes de calor ni condiciones de contorno.

Esta clase de razonamientos son los que dan lugar a las ecuaciones diferenciales más conocidas, pero en algunos casos también se pueden derivar a partir de datos, utilizando técnicas como la expuesta en [16]. Algunas de las ecuaciones diferenciales más conocidas son:

- La ecuación de Laplace $\Delta u = 0$, describe la ley de la conservación en un campo potencial, como el gravitatorio.
- La ecuación de ondas $u_{tt} - \Delta u = 0$, describe el comportamiento de ondas como las formadas por el sonido.
- Las ecuaciones de Navier-Stokes, un sistema de ecuaciones diferenciales que describen el comportamiento de un fluido newtoniano.

Si una ecuación diferencial está descrita mediante derivadas parciales (pues su función solución es una función de varias variables), se denomina ecuación en derivadas parciales, y se suele denotar por sus siglas en inglés "PDE". Si la función solución tiene únicamente una variable, se denomina ecuación diferencial

ordinaria, y se suele denotar por las siglas en inglés "ODE". Esta nomenclatura se adoptará a partir de este punto.

2.2. Métodos clásicos de resolución de ecuaciones diferenciales.

Una vez obtenida la ecuación diferencial que describe un fenómeno, es necesario encontrar soluciones a esta. Una solución de una ecuación diferencial es una función que cumple dicha ecuación en todo punto, y que por tanto nos da el valor del fenómeno en cada punto. Así, si por ejemplo estamos hablando de la ecuación del calor (2.2) su solución u es una función matemática que cumple dicha ecuación y que nos devuelve la temperatura del sistema en un punto y momento dados. En algunos casos se pueden encontrar soluciones simbólicas (es decir, funciones matemáticas), pero en otras es necesario construirlas mediante métodos numéricos. Vamos a ver en qué consisten ambos métodos.

2.2.1. Solución simbólica.

Para ilustrar el funcionamiento de estas soluciones, vamos a mostrar cómo se resuelve la ecuación del calor (2.2) siguiendo los pasos de [2]. Dado que el proceso es algo complejo, vamos a centrarnos más en las ideas generales que en los propios pasos para obtener la solución.

Antes de continuar, vamos a indicar qué significan algunos operadores y notaciones que serán utilizados en esta sección:

- Derivada parcial $\frac{\partial u}{\partial x_i}$: En una función de varias variables $u(x_1, \ldots, x_n)$, se trata de la derivada respecto de una de ellas, indicando cual mediante el numerador de la fracción que compone el símbolo. A menudo se escribe $\frac{\partial u}{\partial x_i} = u_i$ y $\frac{\partial}{\partial x_i}\frac{\partial u}{\partial x_i} = \frac{\partial^2 u}{\partial x_i^2} = u_{ii}$.
- Gradiente ∇u: Es el **vector** resultante de calcular la derivada parcial respecto de cada una de las variables de u. Por tanto $\nabla u = (u_{x_1}, \ldots, u_{x_n})$.
- Derivada total Du: Es la **función** resultante de calcular la suma de las derivadas parciales respecto de todas las variables de u. Por tanto $Du = u_{x_1} + \ldots + u_{x_n}$
- Operador de Laplace Δ: Es la función resultante de calcular la suma de las derivadas parciales segundas respecto de todas las variables de u. Por tanto $\Delta u = u_{11} + \ldots + u_{nn}$

En primer lugar, a la hora de obtener una solución a una ecuación diferencial, nos interesa conocer sus propiedades (que, como norma general, son particulares a cada ecuación). En este caso, vamos a suponer que existe una función $u(x,t)$ que resuelve la ecuación. De ser así, entonces la función $u(\lambda x, \lambda^2 t)$, también la resuelve:

$$\Delta u(\lambda x, \lambda^2 t) = \lambda^2 \sum_{i=1}^{n} u_{ii}(\lambda x, \lambda^2 t) = \lambda^2 u_t(\lambda x, \lambda^2 t) = \frac{\partial}{\partial t} u(\lambda x, \lambda^2 t) \tag{2.3}$$

Esta propiedad de la ecuación indica que la proporción entre x y t es importante. Concretamente si tomamos $r = ||x|| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$, parece obvio que la proporción que nos interesa será $\frac{r^2}{t}$. La manera de interpretar esta propiedad es la siguiente: si toda solución u de la ecuación tiene esta propiedad, los parámetros x y t podrían estar interactuando internamente en la función, de manera que los valores de las λ se simplifiquen. Es decir, si tomamos una función de dos parámetros, pero con la siguiente forma: $u(x,t) = v\left(\frac{x}{t^2}\right) = v(y)$, esta podría ser una candidata a solución, pues cumple la condición que hemos visto.

Conocida esta propiedad, el siguiente paso es explotarla para reducir la complejidad del problema. En este caso la idea es jugar con el valor de λ para conseguir una solución con la forma que hemos comentado. Partiendo de la solución $u(\lambda^\beta x, \lambda t)$, vamos a tomar $\lambda = t^{-1}$, con lo que nos queda una función del tipo $v(\lambda^\beta x)$, donde λ es una función de t, es decir, $\lambda = \lambda(t)$. Esto va a afectar al comportamiento de las derivadas respecto de x y de t, por lo que vamos a considerar como función candidata al producto $\lambda^\alpha \cdot v(\lambda^\beta x) = \frac{1}{t^\alpha} v\left(\frac{x}{t^\beta}\right)$.

Volviendo a la ecuación diferencial, sustituyendo por nuestra solución candidata resulta:

$$\Delta\left(\frac{1}{t^\alpha} v\left(\frac{x}{t^\beta}\right)\right) - \frac{\partial}{\partial t}\left(\frac{1}{t^\alpha} v\left(\frac{x}{t^\beta}\right)\right) = 0 \tag{2.4}$$

$$t^{-(2\beta+\alpha)} \Delta v\left(\frac{x}{t^\beta}\right) - \left(-\beta t^{-(\alpha+1)} t^{-\beta} x Dv\left(\frac{x}{t^\beta}\right) - \alpha t^{-(\alpha+1)} v\left(\frac{x}{t^\beta}\right)\right) = 0 \tag{2.5}$$

$$\tag{2.6}$$

Tomando ahora $y = xt^{-\beta}$ resulta

$$t^{-(2\beta+\alpha)} \Delta v(y) + \beta t^{-(\alpha+1)} y Dv(y) + \alpha t^{-(\alpha+1)} v(y) = 0 \tag{2.7}$$

Ahora, para reducir la ecuación a una incógnita, tomatmos $\beta = \frac{1}{2}$

$$t^{-(\alpha+1)}\left(\Delta v(y) + \frac{y}{2}Dv(y) + \alpha v(y)\right) = 0 \tag{2.8}$$

$$\Delta v(y) + \frac{y}{2}Dv(y) + \alpha v(y) = 0 \tag{2.9}$$

Ahora vamos a asumir que la solución a la ecuación diferencial es radial. Esto se puede explicar de una manera más intuitiva: el calor se desplaza por igual en todas las direcciones, por tanto dado un foco de calor en un punto, su influencia sobre los puntos que le rodean depende de la distancia a estos que de en qué dirección se encuentren. Por tanto existe una función $w(||y||) = w(r)$ con $r = r(y) = \sqrt{y_1^2 + y_2^2 + \ldots + y_n^2}$ (no olvidemos que por ser $y = t^{-1/2}x = t^{-1/2}(x_1, x_2, \ldots, x_n)$, y es un vector), de manera que $v(y) = w(||y||) = w(r)$. Sustituyendo en 2.9 resulta:

$$\Delta w(||y||) + \frac{y}{2}Dw(||y||) + \alpha w(||y||) = 0 \tag{2.10}$$

$$\Delta w(r(y)) + \frac{y}{2}Dw(r(y)) + \alpha w(r(y)) = 0 \tag{2.11}$$

Ahora hay que tener en cuenta las siguientes igualdades:

$$Dr = \frac{y}{r} \quad \text{y} \quad \Delta r = \frac{n-1}{r} \tag{2.12}$$

$$\Delta w(r) = w''(r) \cdot (Dr)^2 + w'(r)\Delta r \tag{2.13}$$

$$Dw(r) = w'(r)Dr \tag{2.14}$$

$$y \cdot y = y^2 = r^2 \tag{2.15}$$

Aplicando esto en (2.11) resulta:

$$w''(r)\,(Dr)^2 + w'(r)\Delta r + \frac{y \cdot y}{2r}w'(r) + \alpha w(r) = 0 \tag{2.16}$$

$$w''(r)\frac{r^2}{r^2} + \frac{n-1}{r}w'(r) + \frac{r}{2}w'(r) + \alpha w(r) = 0 \tag{2.17}$$

$$w''(r) + \frac{n-1}{r}w'(r) + \frac{r}{2}w'(r) + \alpha w(r) = 0 \tag{2.18}$$

Que es una ecuación diferencial de una única variable, con lo que el problema ha quedado bastante simplificado. Como todas las derivadas son respecto de una

única variable, a partir de aquí vamos a escribir $\frac{d}{dr}w = w'$. Ahora vamos a tomar $\alpha = \frac{n}{2}$, y vamos a multiplicar en ambos miembros por r^{n-1}. Reordenando en 2.18

$$(n-1)r^{n-2}w' + r^{n-1}w'' + \frac{1}{2}\left(nr^{n-1}w + r^n w'\right) = 0 \tag{2.19}$$

$$\left(r^{n-1}w'\right)' + \frac{1}{2}\left(r^n w\right)' = 0 \tag{2.20}$$

$$\left(r^{n-1}w' + \frac{1}{2}r^n w\right)' = 0 \tag{2.21}$$

Que es una ecuación diferencial respecto de r. Integrando resulta:

$$r^{n-1}w' + \frac{1}{2}r^n w = a. \tag{2.22}$$

Para cierta $a \in \mathbb{R}$. Aquí se aplica un razonamiento algo complejo en el que no profundizaremos, que permite justificar que $a = 0$. Por tanto, despejando w' resulta:

$$w' = -\frac{1}{2}rw. \tag{2.23}$$

Que es una ecuación diferencial de una única variable cuya solución es:

$$w = e^{-\frac{r^2}{4}+c} = be^{-\frac{r^2}{4}} \tag{2.24}$$

para cierta $b \in \mathbb{R}$. Con esto no hemos acabado aún. Tenemos que incluir las consideraciones de α y β que hemos tomado para ajustar la ecuación planteada en 2.4. Con ello obtenemos la solución a la ecuación del calor (2.2):

$$u(x,t) = \frac{b}{t^{n/2}}e^{-\frac{||x||}{4t}} \tag{2.25}$$

Es claro que el proceso de obtención de una solución es bastante complejo. Lo es más aún si consideramos que esta solución sólo es aplicable en algunos casos específicos, y que la solución general viene dada por una fórmula mucho más extensa. Además, en cada caso es necesario ajustar las constantes de la ecuación, lo cual añade aún más complejidad al proceso.

En definitiva, el método de resolución simbólico sólo es realmente útil en los casos en los que el problema es suficientemente simple, o existe un método para llegar a la solución que simplifique el proceso.

2.2.2. Resolución mediante métodos numéricos.

Los métodos numéricos toman un enfoque distinto al simbólico. En este caso, el objetivo es obtener una función (o un algoritmo) que devuelva valores aproximados, y si es posible, dar una estimación del error cometido. Si bien estos métodos cometen en todos los casos cierto error, son muy útiles para aproximar soluciones demasiado complejas como para ser obtenidas simbólicamente.

En este caso para la experimentación se ha utilizado el Método de los Elementos Finitos (MEF, o FEM por sus siglas en inglés), por lo que nos centraremos en las características de este. Dado que la teoría detrás de esta técnica es bastante compleja, y una explicación completa sería demasiado extensa, únicamente se dará una idea general de su funcionamiento.

La aplicación del MEF a un problema tiene tres pasos:

- La construcción de la formulación débil de la ecuación diferencial del problema.
- La construcción de un mallado sobre la región sobre la que está planteada la ecuación diferencial.
- La aplicación de métodos numéricos para el cálculo y representación de la solución de la ecuación.

Vamos a explicar brevemente en qué consiste cada uno de estos pasos. Para ello, vamos a servirnos de un ejemplo para ilustrar el proceso. En este caso, se ha elegido otro problema relacionado con el calor. Consiste en una placa Ω que comienza el experimento a temperatura 0, y está aislada en los lados derecho e izquierdo. La temperatura exterior es de 5, y el calor fluye a lo largo de los bordes superior e inferior hacia la placa a lo largo del tiempo. Pasados 5 segundos se enciende un foco de calor en el centro de la placa que continúa encendido hasta el final del experimento, a los 10 segundos. La descripción de este problema en forma de sistema de ecuaciones diferenciales (necesaria para construir un modelo numérico) sería la siguiente:

$$\begin{cases} u_t - \Delta u = f & \text{Sobre} \quad \Omega & \text{(2.26a)} \\ \nabla u \cdot \nu = 0 & \text{Sobre} \quad \Gamma_1 & \text{(2.26b)} \\ \nabla u \cdot \nu = u - 5 & \text{Sobre} \quad \Gamma_2 & \text{(2.26c)} \end{cases}$$

Donde Ω representa el subconjunto de $\mathbb{R}^2$ que representa la placa, y Γ_1 y Γ_2 las fronteras de Ω, siendo la primera la pareja de bordes aislados, y la

segunda la pareja de bordes sin aislamiento respectivamente. Los productos $\nabla u \cdot \nu$ representan la componente del gradiente de la temperatura que tiene dirección normal al borde sobre el que está definida.

En primer lugar, veamos cómo se obtiene la formulación débil de este problema. La formulación débil de una ecuación diferencial consiste en la reformulación de dicha ecuación como una ecuación integral (en la que ambos miembros de la ecuación están compuestos únicamente por integrales), y con la adición de las llamadas funciones "test". La teoría necesaria para explicar el proceso correctamente es muy extensa, por lo que daremos una idea general de las justificaciones de cada paso. Partimos en primer lugar de la ecuación del calor (2.26a). Supongamos que tenemos una función u que la cumple. En ese caso, la ecuación también se cumple si multiplicamos en ambos miembros por otra función v cualquiera:

$$u_t \cdot v - \Delta u \cdot v = f \cdot v \tag{2.27}$$

Y por tanto, si integramos sobre todo el dominio Ω, esta igualdad debería seguir cumpliéndose:

$$\int_\Omega u_t \cdot v \, dS - \int_\Omega \Delta u \cdot v \, dS = \int_\Omega f \cdot v \, dS \tag{2.28}$$

En su forma actual, se puede remarcar que la función test v que añadimos previamente nos garantiza que la ecuación diferencial inicial se cumple sobre todo Ω, pues si no estuviese incluida, no tendría por qué ser así. Esto se debe a que la función test puede ser cualquier función integrable definida en Ω, por lo que la igualdad debe cumplirse en cualquier subconjunto medible de Ω.

Este último paso de integración nos permite utilizar un marco teórico con el cual podemos seguir avanzando. Aplicando el conocido como Teorema de Green llegamos a la siguiente ecuación:

$$\int_\Omega \frac{\partial u}{\partial t} \cdot v \, dx \int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Gamma (u - 5) \cdot v \, dS + \int_\Omega f \cdot v \, dx \tag{2.29}$$

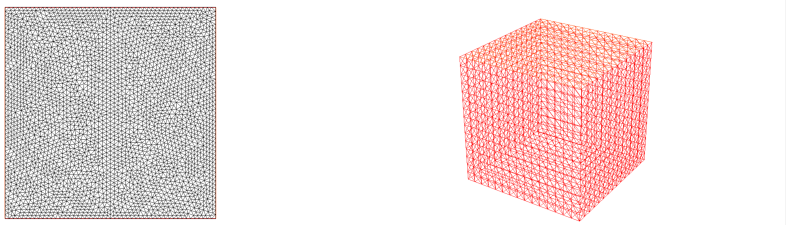
Esta sería la forma final de la formulación variacional del problema. No obstante, consideramos al tiempo como una variable especial, por lo que la utilizaremos de un modo distinto. Vamos a reescribir $\frac{\partial u}{\partial t}$ como la diferencia finita $\frac{u^{n+1} - u^n}{dt}$, donde u^n es el conjunto de valores de u sobre la placa en la iteración n, y u^{n+1} su análogo en la iteración $n + 1$. El valor dt sería un valor

fijo que representa el salto temporal entre n y $n+1$. De este modo, el tiempo de la iteración n es $n * dt$. Por tanto la formulación final del problema sería:

$$\int_\Omega \frac{u^{n+1} - u^n}{dt} \cdot v \, dx \int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Gamma (u - 5) \cdot v \, dS + \int_\Omega f \cdot v \, dx \qquad (2.30)$$

Hay que tener en cuenta que al utilizar esta formulación, podremos hallar soluciones del problema a lo largo del tiempo de manera iterativa. En primer lugar asumiríamos que conocemos u^0, y con ello resolveríamos u^1. Repitiendo este paso se pueden obtener todas las soluciones del problema a lo largo del tiempo.

Hablemos ahora del mallado de un problema. A la hora de representar un fenómeno (independientemente de la técnica usada), siempre es necesario construir o consderar un espacio en el que dicho fenómeno se desarrolle. En el caso del MEF, se construye una estructura conocida como mallado (ver Figura 2.1).



(a) Mallado para el MEF de un cuadrado. (b) Mallado para el MEF de un cubo.

Figura 2.1: Dos ejemplos de mallado para MEF

Estos mallados nos permiten plantear la formulación débil que hemos visto como un sistema de ecuaciones simples, pero de gran tamaño. En líneas generales lo que se hace es plantear en cada nodo (punto del mallado en el que convergen varios segmentos) una ecuación algebraica obtenida a partir de una serie de teoremas cuyo contenido se escapa del ámbito de este texto. Estas ecuaciones planteadas en cada nodo se combinan para obtener el sistema de ecuaciones que comentamos antes.

Por último queda hablar de los métodos numéricos utilizados para el cálculo final de la solución. En primer lugar, sobre el sistema de ecuaciones que planteamos en el paso anterior se aplica un método de resolución numérica que aproxima la solución, con lo que obtenemos una aproximación bastante fiable del valor de u en cada nodo del mallado. Dado que para construir una representación el conocimiento de estos valores no es suficiente (salvo que el mallado

sea suficientemente denso), se aplica un método de interpolación numérica para obtener los valores en cualquier punto y se usa la salida de esta interpolación para representar y obtener resultados numéricos de la simulación en cualquier punto.

2.3. Aplicación de redes neuronales a la simulación de fenómenos físicos.

En los últimos años, con la llegada de nuevas técnicas de construcción y entrenamiento de redes neuronales, se ha abierto una nueva posibilidad para la generación de simulaciones fiables de fenómenos físicos. Mediante ciertas técnicas de entrenamiento, y con una cantidad suficiente de datos, se pueden dar predicciones o construir simulaciones de dichos fenómenos. La ventaja principal del uso de redes neuronales radica en que la construcción de estos modelos resulta mucho menos compleja para el usuario. Por otro lado, los resultados son a menudo menos exactos, y es necesaria una capacidad de computación mucho mayor.

Las redes neuronales artificiales son un paradigma de construcción de modelos de regresión o clasificación basado ligeramente en las redes de neuronas presentes en la mayoría de seres vivos. La estructura de una red neuronal se suele representar mediante un grafo dirigido en el que los nodos representan las neuronas y las aristas sus axones. El gráfico 2.2 muestra un ejemplo de representación.

A la hora de realizar una predicción, cada uno de los nodos que componen la red neuronal realiza un cálculo en base a las señales que le llegan de las aristas entrantes a este. En función del resultado, el nodo se "dispara" y devuelve una salida numérica. El resultado de este proceso, repetido a lo largo de todas las neuronas de la red, da lugar a una predicción.

Esta idea resulta muy versátil a la hora de construir modelos de regresión (y también de clasificación). La versatilidad se basa en la innumerable cantidad de estrategias de entrenamiento y estructuras distintas que se le pueden dar a una red neuronal. La implementación más simple y extendida de estas consiste en dar una estructura de red neuronal como la del ejemplo anterior, y entrenarla sobre un conjunto de entrenamiento clásico (Datos, Valores de Salida/Etiquetas) utilizando las técnicas conocidas como "Descenso de Gradiente" y "Backpropagation". Vamos a comentar brevemente cómo se aplicaría esta implementación simple a la predicción de valores de magnitudes físicas.

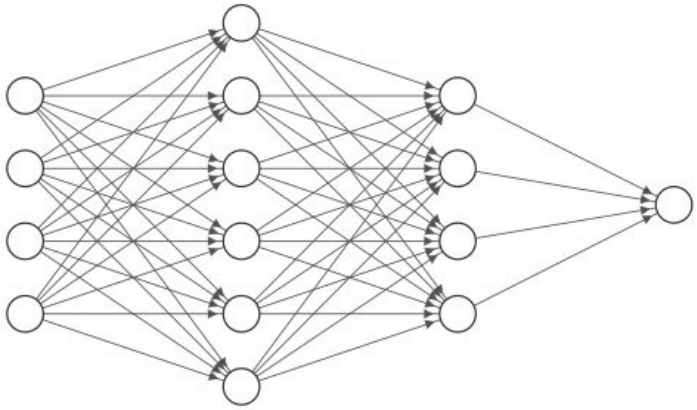


Figura 2.2: Representación de una red neuronal densa (DNN).

2.3.1. Simulación de fenómenos físicos con redes neuronales clásicas.

Supongamos que tenemos un fenómeno físico del que hemos obtenido una cantidad razonable de datos. Por ejemplo, si se trata de predecir la temperatura en una habitación, necesitaremos haber tomado medidas de la temperatura en distintos puntos de dicha habitación en distintos tiempos. Tomando este conjunto de datos compuesto por las mediciones, y alguna parametrización de los lugares y tiempos en que han sido tomadas, podemos generar una DNN como la mostrada en la Figura 2.2, que prediga la temperatura en un punto de la habitación. También es posible utilizar redes neuronales con una arquitectura más compleja para intentar obtener mejores resultados, como por ejemplo redes neuronales convolucionales, o redes neuronales recurrentes.

Sin importar qué arquitectura se utilice, esta clase de uso de redes neuronales para la predicción de comportamientos físicos tiene varias limitaciones muy claras. En primer lugar, la cantidad de datos disponibles no siempre es suficiente en cantidad y densidad. Esto quiere decir que la cantidad de sensores disponibles para medir un fenómeno puede no ser suficiente, y estos pueden estar repartidos de manera que no sean capaces de recabar datos en aquellas zonas en las que se produce la información más relevante. Por poner un ejemplo, se puede medir la temperatura en las paredes de un alto horno, pero no en su interior, que es donde está localizado el foco principal de calor.

En segundo lugar, los datos disponibles no tienen por qué estar libres de mediciones erróneas o poco precisas. Esto implica que si no se dispone de una gran cantidad de datos, seguramente el modelo que construyamos esté mal ajustado debido a la contaminación en los datos.

En definitiva, este método de entrenamiento a menudo dará resultados imprecisos en los casos en que el fenómeno a predecir sea complejo o la cantidad de información disponible sea demasiado pequeña o esté demasiado sesgada. Además, de este modo no se hace uso del conocimiento de que disponemos hoy en día sobre muchos fenómenos físicos, codificado en ecuaciones diferenciales como las presentadas en Sección 2.1.

En la experimentación que se ha realizado se ha utilizado un tipo de red neuronal llamada "Phyisics-Informed Neural Network" (PINN), caracterizada no tanto por su estructura sino por su técnica de entrenamiento. Este tipo de red neuronal sí que permite combinar los datos obtenidos de sensores y mediciones, y el conocimiento codificado en forma de ecuación diferencial. Vamos a explicar en qué consiste y de qué modo se ha implementado.

2.3.2. Physics-Informed Neural Networks (PINNs).

Concepto.

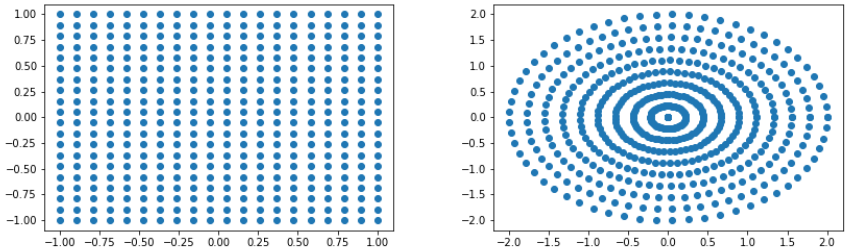
Como ya hemos visto, a la hora de construir un modelo que simule un fenómeno físico, el objetivo es obtener una función (o funciones) $u(X, t)$ que nos devuelva valores (o vectores) que equivalgan a magnitudes medibles, como la temperatura, la dirección y velocidad del viento, etcétera. La función objetivo tiene que cumplir ciertas normas físicas, obtenidas a través de experimentación o distintos razonamientos, y codificadas en una ecuación diferencial.

Las PINNs [15] son redes neuronales cuyo método de entrenamiento nos permite utilizar esta información codificada en ecuaciones diferenciales a modo de datos de entrenamiento. De este modo, se puede evitar la necesidad de gran cantidad de mediciones efectuadas a través de sensores que necesitaría una red neuronal convencional para ser entrenada. Además, también abren una nueva vía en la metodología de interpretación de datos. Esto se debe a que, en el caso de haber disponible una cantidad suficiente de datos, las PINNs pueden ser utilizadas para descubrir las ecuaciones diferencias que rigen el sistema del que provienen.

Espacio de datos.

A la hora de simular un fenómeno físico, hay que tener en cuenta que éste se suele simular sobre un espacio determinado, formado por un objeto o un conjunto de ellos. Por ejemplo, podemos estar interesados en simular el comportamiento del aire alrededor del fuselaje de un avión en movimiento, el calor transferido al ambiente en un disipador, o la distribución de cargas en una viga. Por tanto, a la hora de generar una simulación por computador, el primer paso suele ser la construcción de dicho espacio en el que se desarrolla la simulación o predicción.

En el caso de las PINNs, este espacio consiste en un conjunto de puntos distribuidos a lo largo del espacio a simular. Estos puntos pueden estar distribuidos de manera uniforme, formando un mallado o una cuadrícula, o también de manera irregular, variando la densidad de puntos en cada región. En el segundo caso, normalmente se incrementa la densidad en aquellas regiones en las que se sabe a priori que la función objetivo presenta muchas variaciones (por ejemplo en un foco de calor), y se reduce la densidad en aquellas que hay poca variación (para reducir el coste computacional). En la Figura 2.3 se muestran dos ejemplos de mallados para PINN.



(a) Mallado uniforme centrado en (0,0) (b) Mallado radial centrado en (0,0)

Figura 2.3: Dos ejemplos de mallado para PINN

Este conjunto de puntos es importante a la hora de entender el método de entrenamiento. A partir de ahora, denotaremos el espacio simulado como (X, T), donde X representa las dimensiones espaciales y T la temporal. Al conjunto de puntos que utilizan las PINNs lo denotaremos como (X^*, T^*), y cada punto de dicho conjunto se escribirá como (x^*, t^*).

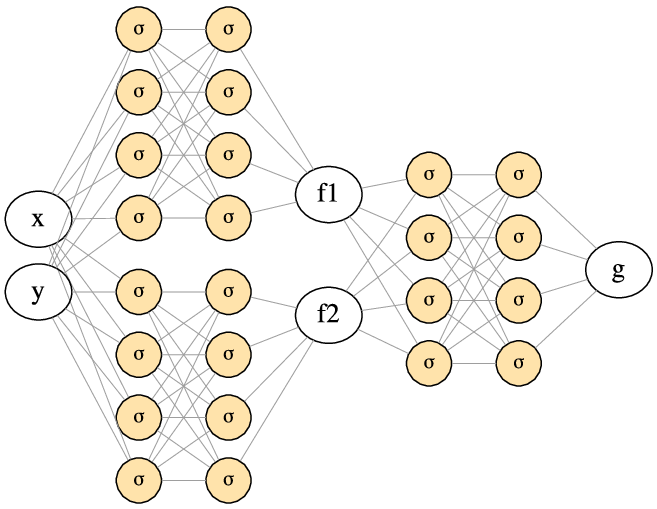


Figura 2.4: Ejemplo de estructura de PINN mostrada en [6].

Arquitectura.

Una vez construido el espacio simulado, se puede pasar a construir la PINN. La estructura de la PINN será la de una red neuronal que tenga por entradas los puntos de (X^*, T^*) y por salidas el valor (o valores) a predecir en cada uno de estos puntos. La teoría detrás de las PINNs hace posible utilizar una cantidad de estructuras muy variada: Red Neuronal Densa (DNN), Red Neuronal Convolucional (CNN), Long-Short Term Memory (LSTM), etc.

En la Figura 2.4 se muestra una posible estructura de una PINN propuesta en [6], en la que se pretende imitar a la estructura de la función $g(x, y) = g(f_1(x, y), f_2(x, y))$.

En los casos en los que intervienen varios elementos a predecir conviene construir varias PINN en vez de una. Un ejemplo de esto son las ecuaciones de Navier-Stokes, en las que se intenta predecir la velocidad, aceleración y presión del fluido en cada punto (entre otras magnitudes). En casos como este, las salidas de cada red neuronal se unen mediante capas Lambda que permiten aplicar las operaciones relacionadas con ecuaciones diferenciales, para así poder calcular el descenso de gradiente y el backpropagation apropiadamente.

Entrenamiento para modelado de fenómenos físicos.

Vamos a hablar ahora de la técnica de entrenamiento de la PINN. Como norma general, se puede incluir información en forma de datos, es decir, asociar a un punto (x^*, t^*) un valor específico u^*. No obstante, la mayoría de la información proviene de la ecuación o ecuaciones diferenciales utilizadas en el entrenamiento. Esto se hace reformulando dichas ecuaciones como funciones cuyo dominio son las funciones u predichas (dadas por redes neuronales) y cuya imagen es $\mathbb{R}$. En base a la salida de cada función se calcula el coste o "loss" de la red neuronal. Vamos a explicar de una manera más específica cómo se hace esta reformulación y cómo se utiliza.

Supongamos que tenemos un conjunto de ecuaciones diferenciales cuya incógnita es la función u (también se pueden formular usando varias funciones incógnita). En ese caso tenemos varias ecuaciones del tipo:

$$G(u) = F(u) \qquad \text{Sobre (X,T)} \tag{2.31}$$

Por ejemplo si tenemos la ecuación del calor (2.2), entonces (escribiéndola de manera muy poco ortodoxa) tenemos que

$$G(*) = \int_{(X,T)} \frac{\partial *}{\partial t} dV,$$
$$F(*) = \int_{(X,T)} \sum_{i \in I} \frac{\partial^2 *}{\partial x_i^2} dV.$$

No nos interesa contar con dos funciones en una ecuación, sino con una única función igualada a cero. Por tanto reformulamos la ecuación (2.31) del siguiente modo:

$$H(u) = G(u) - F(u) = 0 \qquad \text{En (X,T)}$$

Para el caso de las PINNs, dado que estamos trabajando sobre un mallado (X^*, T^*), no es necesario calcular ninguna integral. Sólo es necesario evaluar la restricción en cada punto del mallado y agregar los resultados como un sumatorio. Para adaptar la notación, escribiremos $H(u(x^*, t^*))$ como la evaluación de la restricción H sobre la función u, únicamente sobre el punto (x^*, t^*).

Combinando todas las funciones asociadas a cada restricción y los valores de u conocidos, se puede computar un valor de pérdida en cada punto. Supongamos que tenemos un mallado (X^*, T^*), y que hemos construido una red neuronal

$N_u(W;x,t) = u^* \approx u$, donde $N_u(W;x,t)$ representa una red neuronal que aproxima la solución u, y W representa los pesos actuales de la red neuronal. La función u^* es la solución dada por la red neuronal (entendida como función). En ese caso, si tenemos un conjunto de restricciones $H_i(u), i \in I$ y un conjunto de puntos (x^*, t^*) en los que conocemos el valor de u, la función pérdida podría venir dada como:

$$\mathcal{L}(W) = \overbrace{\sum_{i \in I} \left(\sum_{(x^*,t^*) \in (X^*,T^*)} ||H_i(u^*(x^*,t^*))|| \right)}^{\text{Información relativa a PDEs}}$$

$$+ \overbrace{\sum_{(x^*,t^*) \in (X^*,T^*)} ||u(x^*,t^*) - u^*(x^*,t^*)||}^{\text{Información relativa a datos disponibles}}. \tag{2.32}$$

Esta función es la base principal del entrenamiento de PINNs para modelaje de fenómenos físicos. Utilizándola como loss, se puede implementar el descenso de gradiente para realizar el entrenamiento en redes neuronales.

Descubrimiento de ecuaciones diferenciales.

Otro posible uso de las PINNs es el descubrimiento de ecuaciones diferenciales. Este planteamiento alternativo es muy útil en el sentido de la interpretación de un conjunto de datos, pues las ecuaciones diferenciales, aunque son algo más complejas de interpretar, permiten expresar relaciones de dependencia entre variables más complejas que, por ejemplo, las relaciones lineales que se aprecian con un regresor lineal. Aunque esta aplicación no es el objetivo de este trabajo, vamos a comentarla brevemente.

En los casos en los que hay disponible gran cantidad de datos, el proceso que comentamos en la Sección 2.3.2 se puede modificar levemente para obtener relaciones entre datos dadas por ecuaciones diferenciales. En [15] se detalla un ejemplo de esto. En el artículo se recuperan las ecuaciones diferenciales que rigen los datos obtenidos de un experimento propuesto por ellos mismos.

Para hacerlo, los autores asocian a cada uno de los diferenciales que componen la ecuación un parámetro λ_i , el cual también es ajustado durante el proceso de entrenamiento. Una vez terminado el entrenamiento, se puede recuperar el valor de todos los parámetros incluidos, y ver si estos concuerdan con los de la

ecuación de la que se obtuvieron los datos del experimento. En el artículo, los autores obtuvieron tasas de ajuste muy altas, incluso en los casos en los que se añadió ruido a los datos de entrenamiento.

2.4. Estado del arte en la aplicación de redes neuronales a la simulación de fenómenos físicos.

A día de hoy existen varias herramientas que facilitan la implementación de PINNs. Aunque estas herramientas aún se encuentran en fases iniciales del desarrollo, y existen limitaciones en cuanto a los tipos de estructura de red neuronal disponibles, permiten generar modelos con gran facilidad.

En primer lugar, SciANN [6] ha sido el paquete de Python utilizado en este proyecto durante la fase de experimentación. Este paquete está basado en Tensorflow/Keras, y permite la construcción y entrenamiento de PINNs. Por desgracia, no cuenta con módulos de construcción de mallados ni de visualización de resultados, por lo que estas tareas se han de realizar con otros paquetes, en este caso NumPy [8] y MatPlotLib [11].

Por otro lado, el framework de NVIDIA, llamado NVIDIA-SimNet [10] es una implementación más potente y versátil, aunque de uso más complejo. Este framework combina técnicas de generación de mallados, resolución de ecuaciones diferenciales mediante redes neuronales, y gran cantidad de opciones como distintas funciones de pérdida, métodos de resolución especializados para las ecuaciones de Navier-Stokes y métodos de visualización de resultados.

Para finalizar este capítulo, cabe destacar que los dos métodos comentados no son las únicas técnicas disponibles para simulación de físicas basadas en redes neuronales existentes. Por ejemplo, en [13], se propone el uso de GNNs (Graph Neural Networks) para la simulación de físicas a través de mallados conexos (aquí entendidos como grafos) como los que se usan en el MEF. En este artículo los autores consiguen obtener buenos resultado modelando mecanismos tan complejos como una bandera ondeando en el viento.

También existen propuestas para el modelaje de problemas que no pueden ser descritos únicamente mediante PDEs. En algunas ocasiones las ecuaciones que rigen el sistema que se pretende simular incluyen cálculo integral, para el cual las PINN no están preparadas pues se apoyan en el cálculo de diferenciales en los puntos dados por los datos de entrenamiento, y no son capaces de agregar los datos para realizar una integral. En el artículo [12] se propone una arquitectura llamada fPINN que hibrida métodos numéricos con la técnica de las PINN para

conseguir resolver esta clase de ecuaciones.

Finalizada esta introducción teórica a los métodos utilizados en la experimentación, vamos a comentar en qué consiste la experimentación, y qué herramientas se han utilizado.

Capítulo 3

Metodología.

En este capítulo se explicará de manera general en qué consiste y con qué herramientas se ha construido el entorno experimental. Este entorno consiste en cuatro problemas teóricos con complejidad creciente, aumentando el número de dimensiones en cada paso.

La estructura de este capítulo es la siguiente: en la Sección 3.1 se da una visión global del proceso de experimentación. En la Sección 3.2 se explica qué técnicas se han usado para implementar los métodos clásicos explicados en el Capítulo 2, los cuales han sido utilizados para obtener los datos de validación de los modelos PINN. En la Sección 3.3 se explica qué pasos sigue el proceso de construcción de una PINN con las herramientas que se han utilizado. Después, en la Sección 3.4 se explica en qué consisten los experimentos propuestos. Por último, en la Sección 3.5 comentaremos qué métricas se han utilizado para evaluar el rendimiento de cada modelo construido.

3.1. Pautas generales de la experimentación.

Como ya se ha comentado previamente, el objetivo de este proyecto es evaluar el rendimiento de las PINN comparándolas con otros métodos disponible. Para hacerlo, se han propuesto cuatro problemas físicos de dificultad creciente expresados mediante sistemas de ecuaciones diferenciales en derivadas parciales. Los enunciados de cada problema están detallados en la Sección 3.4. El objetivo de los tres primeros problemas es realizar una búsqueda sobre una rejilla de parámetros de la PINN, para así poder evaluar qué combinaciones de parámetros dan mejores resultados, y qué problemas en las predicciones son propios de las

PINNs, y ocurren sin importar con qué parámetros se ajusten. A esta búsqueda de parámetros sobre una rejilla y a su posterior análisis es a lo que nos referimos cuando hablamos de experimentación. El cuarto experimento es algo distinto, pues es de una complejidad mucho más alta que los anteriores, y en su caso se han entrenado únicamente dos PINNs utilizando los mejores parámetros de las anteriores experimentaciones, y variando el tipo de datos que se le pasan. De este modo, aunque al cuarto experimento también lo consideramos como parte de la experimentación, su meta es más bien mostrar de manera más clara las limitaciones de las PINNs en un experimento de dificultad más alta.

El proceso de la experimentación tiene dos etapas previas: propuesta de problemas y obtención de datos de validación por medio de métodos clásicos. En cuanto a la propia experimentación, hay varios puntos que conviene explicar.

En primer lugar, se ha considerado una variación sobre 7 parámetros distintos, lo cual hace imposible la experimentación sobre una rejilla compuesta por todos ellos, sobre todo teniendo en cuenta las dimensiones que esta podría alcanzar. Por esta razón, las experimentaciones sobre rejillas se han dividido en 2 fases. En la Subsección 3.4.5 se explica qué parámetros componen la rejilla y de qué manera se han dividido en dos grupos.

El proceso de experimentación se muestra en la Figura 3.1. Vamos a comentar cada paso: en cada fase de cada experimento se genera una lista de posibles variables para cada uno de los parámetros correspondientes a dicha fase, y el resto se configuran con un valor fijo, el cual se elige atendiendo a los resultados del experimento previo (en el caso del experimento 1 se fijaron de manera arbitraria). Después se crea una rejilla de configuraciones formada por todas las combinaciones posibles de parámetros dadas por las variables antes definidas. Después, mediante un bucle, se entrena una PINN con cada una de las configuraciones que componen la rejilla, la cual es evaluada mediante varias métricas, y tanto sus resultados como sus predicciones se guardan. Una vez terminado el bucle de entrenamiento, se obtiene un Dataframe con los parámetros y los resultados en las distintas métricas. Este Dataframe se analiza para ver qué parámetros han dado los mejores resultados, y cuales han dado problemas. Aquella combinación de parámetros con mejores resultados se fija para la siguiente fase o experimento, y aquellos parámetros que resultaran problemáticos se retiran de los siguientes bucles de experimentación.

Para realizar la experimentación se ha utilizado la distribución de Python Anaconda [1], mediante los cuadernos Jupyter que este entorno incluye. La implementación de las PINNs se ha realizado utilizando el paquete SciANN [6], del cual ya hablamos en la Sección 2.4. Para registrar los resultados, parámetros y predicciones de cada modelo a medida que iban siendo entrenados se ha

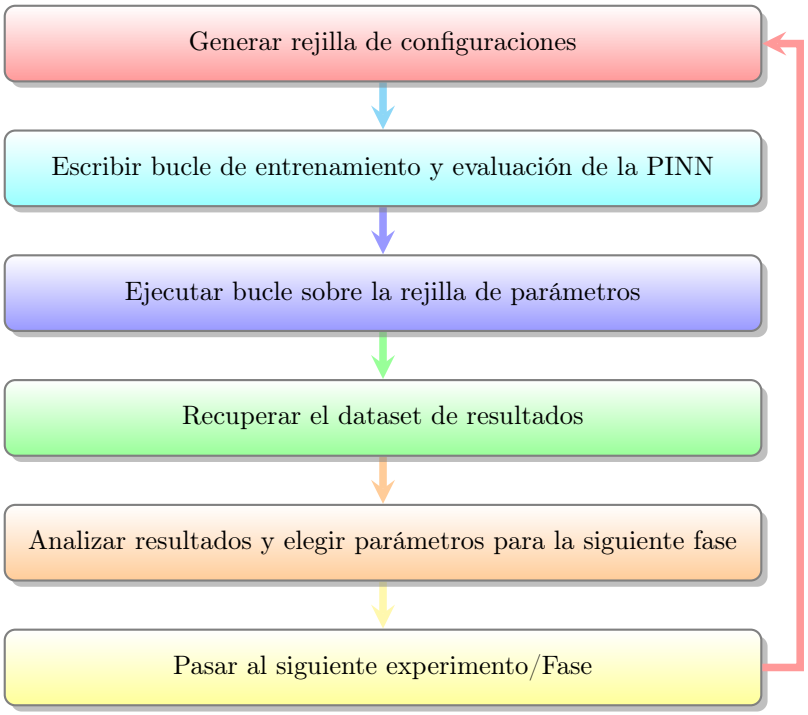


Figura 3.1: Flujo de trabajo en cada fase de la experimentación

utilizado el paquete MLflow, el cual permite registrar y consultar fácilmente tanto los parámetros y métricas relacionadas con cada modelo como elementos mas complejos (gráficas, historiales de entrenamiento, etc.). Para la generación de mallados y la visualización de predicciones ha sido necesario emplear otras bibliotecas, pues SciANN no tiene módulos para ello). En este caso se han utilizado NumPy [8] y Matplotlib [11].

Para analizar los resultados de cada fase de la experimentación, se han utilizado el lenguaje R [14] y el paquete Tidyverse [17]. La razón de esto es que el tamaño de los datasets generados en la experimentación es bastante manejable, y la obtención de gráficos y análisis de datasets es mucho más rápida mediante Tidyverse en estos casos.

Para la obtención de datos de validación se han utilizado Numpy [8] y un software llamado FreeFem++ [9]. El modo de uso de estos dos recursos se detalla en la Sección 3.2.

Por último, dado que se han tomado medidas sobre tiempos de ejecución y se ha tenido en cuenta el coste computacional de la experimentación, es necesario

detallar las características del equipo. Se trata de un ordenador de sobremesa con un procesador Intel Core i5-4460, 16GB de RAM y una tarjeta gráfica NVIDIA GTX 960. El sistema operativo de dicho equipo es un Windows 10 de 64 bits.

.

3.2. Obtención de datos de validación.

Como ya hemos dicho, a la hora de realizar la experimentación es conveniente contar con datos de validación que nos permitan comparar el modelo construido con el modelo que se obtendría utilizando los métodos clásicos. Para el caso del primer experimento se ha empleado una simplificación del método simbólico, y para los siguientes se ha empleado el método numérico.

3.2.1. Método Simbólico.

El método simbólico se ha utilizado únicamente en uno de los casos (el de una única dimensión espacial), y no se ha hecho siguiendo los pasos explicados en el apartado 2.2.1. Esto se debe a que la ecuación del calor completa tiene una gran cantidad de constantes que ajustar, por lo que se ha optado por usar una técnica más simple adaptada al problema en una dimensión explicada en [7], cuya función resultado es la misma que se obtendría con la fórmula completa.

En líneas generales, el método consiste en asumir que la función solución $u(x,t)$ es el producto de dos funciones de una única variable, de modo que $u(x,t) = T(t) \cdot X(x)$. Esto se apoya en los resultados observados en el apartado 2.2.1, pues la función solución hallada tiene estas características.

Utilizando esta suposición y las condiciones de contorno, se describe la función $X(x)$ mediante una serie de Fourier, y aplicando las constantes obtenidas en este último cálculo, se obtiene la serie de $T(t)$. El producto de ambas series término a término da como resultado la serie que describe $u(x,t)$

3.2.2. Método Numérico.

Como ya habíamos comentado, para realizar esta simulación numéricamente se ha utilizado el método de los elementos finitos. La implementación se ha hecho utilizando un software llamado FreeFem++ [9], un software de uso libre escrito en C++. Contiene herramientas que facilitan la generación de mallados y la resolución de EDPs mediante el método de los elementos finitos.

Debido a que no es posible llamar a una experimentación construida en FreeFem++ desde Python para extraer los resultados obtenidos, se ha seguido un camino distinto para producir y almacenar resultados. Utilizando comandos de C++ se han generado dos archivos de texto por cada experimento. El primero contiene la posición de los nodos del mallado (lo cual nos permite construir un mallado al estilo del de una PINN), y el segundo contiene vectores que almacenan los valores del experimento sobre el mallado en cada instante t. Después se ha definido una pareja de funciones en python que permite cargar el mallado y los resultados de cada experimento.

Por último, para obtener los valores de la experimentación en cualquier punto del mallado MEF, y no solo en los nodos de este, se ha utilizado una función de interpolación del paquete SciPy que permite aproximar los valores en puntos intermedios del mallado.

3.3. Aplicación de PINN.

Para construir modelos basados en PINN, se ha utilizado un paquete de Python llamado SciAnn [6], que permite construir y entrenar fácilmente PINN con estructura de Red Neuronal Densa (DNN). SciAnn implementa herramientas para definir todas las variables de un sistema de ecuaciones diferenciales como los que hemos visto en el Capítulo 2, y también para definir las funciones de coste comentadas en la Subsección 2.3.2. Por otro lado, no aporta ningún método para la generación de mallados, por lo que para esto es necesario utilizar otra herramienta. Vamos a comentar en qué consiste el proceso de construcción de una PINN utilizando este paquete:

En primer lugar es necesario construir un mallado (X^*, T^*) que represente el espacio a simular. Como ya hemos dicho, SciAnn no ofrece herramientas para esto, por lo que en este caso se ha hecho mediante el paquete Numpy. Una vez construido el mallado es necesario definir las variables del modelo. En primer lugar se definen tantas variables espaciales como sean necesarias. Por ejemplo si queremos simular una placa (a la cual consideramos un objeto en 2 dimensiones), deberíamos definir dos variables $''x''$ e $''y''$, y por último una variable $''t''$ que simule el tiempo. Estas variables serán los puntos de entrada del mallado al modelo, así que es necesario definirlas antes de construir la función (o funciones) a predecir, "u". Para definir esta función "u" es necesario pasarle una lista con las variables que utilizará como entrada (en el mismo orden que tienen en el mallado), la estructura de la red neuronal en forma de lista de enteros en la que cada entero especifica el número de neuronas en cada capa, y por último

la función de activación que utilizará cada capa.

Terminada la construcción de todas las variables del sistema de ecuaciones, es necesario definir las funciones de coste de cada modelo. Para nuestra experimentación se han utilizado únicamente dos tipos de funciones: funciones de coste que expresan el comportamiento de las magnitudes a predecir en la región interior del mallado, y funciones de coste que expresan condiciones de borde como el aislamiento respecto del entorno.
Por otro lado, en caso de que se quiera incluir información en forma de puntos, es necesario generar una tupla del tipo (puntos, valores) que contenga una lista con los índices de los puntos del mallado a los que se asigna el valor correspondiente en la lista de valores. Después se construye un modelo al que se le pasan las variables y las restricciones (como función y como valor), y otras especificaciones como la función de pérdida y el optimizador a utilizar. Finalmente se entrena el modelo y se obtiene una red neuronal que aproximaría nuestra función "u". En la Figura 3.2 se muestra un ejemplo de montaje de una PINN simple. El código de la PINN del ejemplo se corresponde con la construcción de la PINN que resuelve el problema descrito en la Subsección 3.4.1.

```
#Se definen las variables del problema y la funcion salida "u"
x = sn.Variable('x')
t = sn.Variable('t')
u = sn.Functional("u", [x,t], 3*[100] ,'tanh')

#Se construye el mallado de entrenamiento
xmin,xmax=0,2 # Limites en x e y
tmin,tmax=0,10 # Limites en tiempo t
xrange=xmax-xmin
trange=tmax-tmin

x_data, t_data = np.meshgrid(
  np.linspace(xmin, xmax, xrange*10),
  np.linspace(tmin,tmax, trange*10)
)
x_data = x_data.flatten()[:,None]
t_data = t_data.flatten()[:,None]

#Restricciones
##################
#Calor
calor = sn.constraints.PDE(diff(u,t) - ( (diff(u,x,order=2) )))
L1 = PDE(calor)
datos1 = [('zeros')]
##################

#Datos introducidos como puntos
##################
#TInicial
indInicial = np.where(t_data<TOL)[0]
calorInicial = 2.5*x_data[indInicial,0]

d2= Data(u)
datos2 = [(indInicial[:,None],calorInicial[:,None])]

#Entrada de calor por la dcha
indDerecha = np.where(x_data>2-TOL)[0]
indT_No_Inicial = np.where(t_data>TOL)[0]
indDerecha = np.intersect1d(indDerecha,indT_No_Inicial)[:,None]
calorDerecha = 5

d3 = Data(u)
datos3 = [(indDerecha,calorDerecha)]

#Aislamiento por la izda
indIzda = np.where(x_data>2-TOL)[0]
indT_No_Inicial = np.where(t_data>TOL)[0]
indIzda = np.intersect1d(indIzda,indT_No_Inicial)[:,None]

d4 = Data(diff(u,x))
datos4 = [(indIzda,'zeros')]
##################

#Concatenamos todo
restricciones = L1+d2+d3+d4
datos = datos1+datos2+datos3+datos4

#Definimos el modelo
m = sn.SciModel([x,t], restricciones, 'mae','Adam')

#Entrenamos el modelo
history = m.train([x_data,  t_data], datos ,epochs =500)
```

Figura 3.2: Ejemplo de construcción de una PINN simple

3.4. Experimentos propuestos.

Como ya hemos comentado, se han construido varios experimentos de complejidad creciente para poder conocer en qué medida las PINNs consiguen resultados próximos a los métodos clásicos. En esta sección explicaremos en qué consiste cada uno de los experimentos. En las cuatro primeras secciones hablaremos del problema físico que compone cada uno de los experimentos propuestos, y en la última expondremos el conjunto de parámetros que conforman la rejilla de configuraciones de los tres primeros experimentos.

Antes de pasar a detallar cada experimento, queda comentar que en ninguno de ellos se han considerado magnitudes específicas, es decir, no se expresa el calor en grados centígrados, la longitud en metros o centímetros, ni el tiempo en segundos, minutos u horas. Esto se debe a que el tener en cuenta magnitudes no añade complejidad al problema desde el punto de vista de la construcción del modelo, únicamente la añade desde el punto de vista teórico, lo cual solo implica que los experimentos se vuelven algo más complejos de explicar. Dado que independientemente de las magnitudes se está intentando aproximar una función, esta adición a la complejidad no nos resulta útil ni interesante. Por esta razón, se ha omitido el uso de estas unidades de medida, y se habla de las magnitudes como unidades de tiempo, calor, o longitud.

Otro punto importante antes de continuar es explicar qué información reciben las PINNs durante su entrenamiento. Si bien la técnica de entrenamiento de estos modelos permite incluir datos en cualquier punto y tiempo del experimento, las PINNs que se han construido en esta experimentación sólo han recibido el estado inicial del espacio simulado, las condiciones de contorno del espacio (información en los bordes espaciales), y las ecuaciones que rigen el fenómeno. Todos estos datos serán detallados en forma de sistema de ecuaciones diferenciales en las siguientes secciones. El propósito de este método de uso es utilizar las PINNs del mismo modo en que se emplean los modelos clásicos, los cuales reciben únicamente la información descrita esta clase de sistemas de ecuaciones diferenciales. La única excepción a este método de uso aparece en el experimento 4 (Subsección 3.4.4). Esto se debe a que en la Sección 4.4 se intenta determinar si la exactitud de la PINN mejora aumentando la cantidad de información que recibe.

Vamos a pasar a explicar en qué consiste cada experimento:

3.4.1. Primer Experimento: Calor sobre una barra.

En el primer experimento se intenta simular el calor sobre una barra de 2 unidades de longitud a lo largo de 10 unidades de tiempo. Al principio del experimento cada punto x de la barra se encuentra a temperatura $\frac{5}{2}x$, estando situado el límite izquierdo en $x = 0$ y el límite derecho en $x = 2$. La barra está aislada en el lado izquierdo y en el lado derecho existe una fuente de calor que se mantiene a temperatura 5 a lo largo de todo el experimento. Además, no existe ningún foco o pérdida de calor en el interior de la barra. Con este enunciado, el sistema de ecuaciones que rige el experimento sería:

$$\begin{cases} u_t - \Delta u = 0 & \text{Sobre} \quad \Omega & \text{(3.1a)} \\ \nabla u \cdot \nu = 0 & \text{En} \quad \{x = 0\} & \text{(3.1b)} \\ u = 5 & \text{En} \quad \{x = 0\} & \text{(3.1c)} \end{cases}$$

La solución simbólica a este problema vendría dada por la siguiente serie:

$$u(x,t) = \sum_{n=0}^{\infty} \{exp(-(\frac{2n+1}{4}\pi)^2 t) \cdot \frac{-2.5}{\left(\frac{(2n+1)\pi}{4}\right)^2} cos(\frac{2n+1}{4}\pi x)\} + 5 \qquad (3.2)$$

El aspecto de la función dada por esta serie se muestra en la Figura 3.3

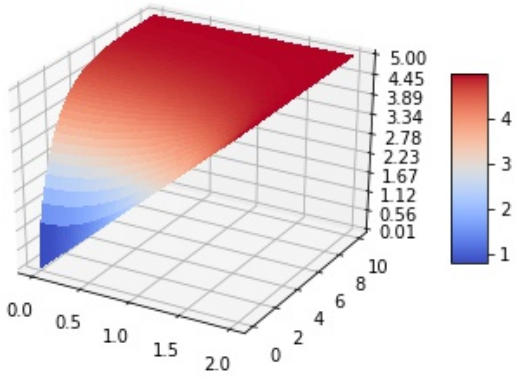


Figura 3.3: Gráfico de la solución simbólica del Experimento 1

Los datos de validación para el problema serán obtenidos de esta función. Dado que se trata de una serie infinita, se ha construido una aproximación mediante Numpy de los primeros 100 elementos de la serie, pues a partir de dicho punto cada elemento de la serie tiene un aporte insignificante a la función. Dado que también es posible realizar una aproximación utilizando el MEF, se ha hecho una implementación cuyos resultados también se tendrán en cuenta en la Subsección 4.1.2.

Por último cabe comentar que dado el aspecto de la función solución, es obvio que este problema es bastante simple. Esto resultará útil para descartar rápidamente distintos parámetros que, como veremos, provocan que el modelo simplemente no sea capaz de conseguir ninguna clase de ajuste aceptable, como por ejemplo un hiperplano.

3.4.2. Segundo Experimento: Foco de calor en una placa.

En el segundo experimento se intenta simular el comportamiento del calor sobre una placa de 2 unidades de ancho por 2 de largo durante 10 unidades de tiempo. En este caso se ha colocado el centro de la placa en $\{x = 0\}$, y sus lados paralelos a los ejes de coordenadas. Cada punto (x, y) de la placa comienza el experimento con temperatura $\frac{5}{2}|y|$, y los bordes $\{y = -2\}$ e $\{y = 2\}$ se encuentran a temperatura ambiente de 5 unidades. Por otro lado, los bordes $\{x = -2\}$ y $\{x = 2\}$ se encuentran aislados del exterior, por lo que no se gana ni pierde calor a través de estos. A los 5 segundos de comenzar el experimento, se enciende un foco de calor en el centro de la placa que continúa encendido hasta el final del experimento. En este caso, el sistema de ecuaciones que rige el experimento sería:

$$\begin{cases} u_t - \Delta u = f & \text{Sobre} \quad \Omega & \text{(3.3a)} \\ \nabla u \cdot \nu = 0 & \text{Sobre} \quad \{x = -2\} \cup \{x = 2\} & \text{(3.3b)} \\ u = 5 & \text{Sobre} \quad \{y = -2\} \cup \{y = 2\} & \text{(3.3c)} \\ u(x, y) = \dfrac{5}{2}|y| & \text{Sobre} \quad \{t = 0\} & \text{(3.3d)} \end{cases}$$

Donde f representa el foco de calor, y está definido por:

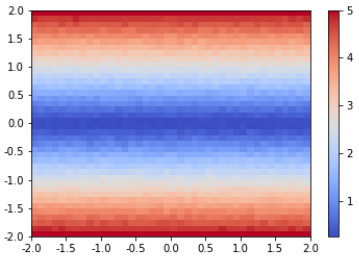
$$f(x, y, t) = \frac{10 \cdot \exp(-\frac{x^2+y^2}{2 \cdot 0.4^2})}{\sqrt{2 \cdot \pi \cdot 0.4^2}} \cdot \frac{1}{1 + \exp(-4 \cdot (t - 6))}$$

La primera fracción de la función representa una distribución gaussiana sobre

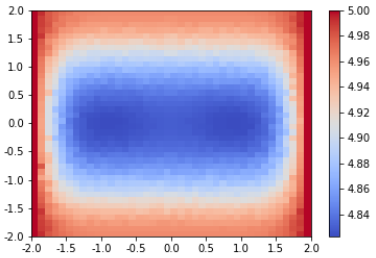
el plano $X \times Y$ con área bajo la curva igual a 10 (es decir, contiene 10 unidades de calor), y la segunda fracción es una función de activación sigmoide que permite simular que el foco de calor se activa en $t = 5$ y aumenta poco a poco hasta llegar a rendimiento completo en torno a $t = 7$.

Tanto la condición de calor inicial $u(x, y) = \frac{5}{2}|y|$ como la función de activación del foco se han implementado de esta manera porque se ha comprobado que las PINN tienen problemas para simular discontinuidades en el entorno experimental, y "empujan" hacia atrás en el tiempo las condiciones que se les aportan en datos futuros. Por ejemplo, si se retira la función de activación del foco de calor, la simulación generada por la PINN la crea artificialmente durante los tiempos previos del experimento. Es conveniente tener este hecho en cuenta si se va a generar alguna simulación a la que se le introduzcan datos no extraídos de un experimento, como es el caso.

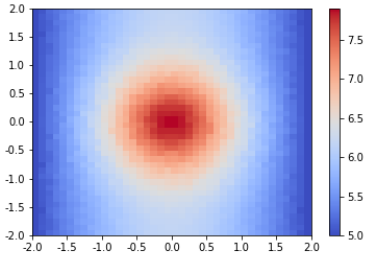
En este caso calcular una solución simbólica se vuelve mucho más complejo por lo que se ha optado por generar una solución numérica utilizando el MEF, y se han utilizado los datos resultantes como datos de validación. Dado que este problema cuenta con 3 dimensiones (2 espaciales y una temporal). No es razonable representarlo como hicimos en el problema anterior. Por ello, en la Figura 3.4 se muestran 3 fases de la evolución del experimento, donde los valores para la representación se ha obtenido a través del MEF.



(a) Experimiento 2 en t=0



(b) Experimiento 2 en t=5



(c) Experimiento 2 en t=10

Figura 3.4: Evolución del experimento 2

En este caso, el número de puntos necesarios para mallar apropiadamente el experimento crece en un orden de magnitud debido al incremento de dimensión. Por tanto durante la experimentación, el mallado de parámetros que evaluaremos será menor que en el anterior.

3.4.3. Tercer Experimento: Pozo de calor en un bloque.

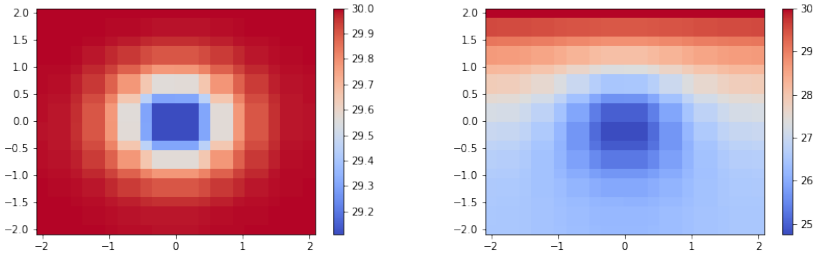
El tercer experimento simula el flujo de calor desde el exterior hacia el interior de un bloque tridimensional con 4 unidades de longitud en cada lado. El bloque está aislado en todas sus caras salvo en la superior, y el exterior se encuentra a una temperatura de 30 unidades, temperatura a la que comienza la totalidad del bloque en el experimento. Desde el inicio de este, se enciende un pozo de calor descrito por la función $f(x, y, z) = -200 \cdot \exp(-\frac{x^2+y^2+z^2}{0.4})$, el cual enfría levemente el bloque a lo largo de las 5 unidades de tiempo que dura este experimento.

El sistema de ecuaciones que rige el experimento sería:

$$\begin{cases} u_t - \Delta u = f & \text{Sobre} \quad \Omega & \text{(3.4a)} \\ \nabla u \cdot \nu = 0 & \text{Sobre} \quad \{\Gamma_1\} & \text{(3.4b)} \\ u = 30 & \text{Sobre} \quad \{\Gamma_2\} & \text{(3.4c)} \\ u(x, y, z) = 30 & \text{Sobre} \quad \{t = 0\} & \text{(3.4d)} \end{cases}$$

Donde Γ_1 es el conjunto de bordes del cubo que están aislados del exterior, y Γ_2 es el borde no aislado del exterior que se encuentra a temperatura constante 30.

En este caso el conjunto de datos de validación se ha obtenido también utilizando el MEF. Debido a la dimensionalidad del problema, no es posible hacer una representación de las fases de este. Por tanto se ha realizado una sección del espacio de simulación en $\{y = 0\}$, y se han representado dos etapas del experimento mostradas en la Figura 3.5



(a) Experimiento 3 en t=0.01 (b) Experimiento 3 en t=5

Figura 3.5: Evolución del experimento 3

Una vez expuestos todos los experimentos que se van a resolver por medio de distintas PINNs, vamos a comentar qué métricas vamos a utilizar para evaluar cómo de apropiadas son las distintas configuraciones a la hora de ajustar una PINN en cada experimento.

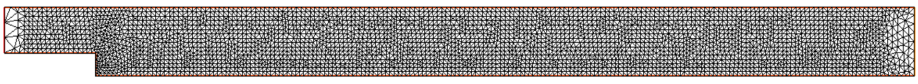
3.4.4. Cuarto Experimento: Tubo de viento en 2 dimensiones.

En el cuarto experimento se ha pretendido aumentar en gran medida la complejidad de la función (en este caso funciones) solución que se pretenden

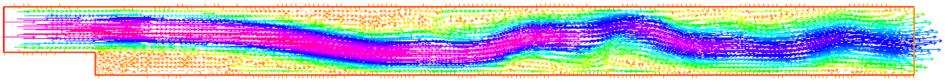
conseguir con el modelo PINN. El objetivo de este problema es mostrar cuanta precisión pierde la PINN al enfrentarse a un problema de complejidad próxima a la de un experimento real, aun utilizando una de las configuraciones que mejores resultados han dado previamente. Por tanto, a diferencia de los otros experimentos, en este no se ha realizado una búsqueda de los mejores parámetros sobre una rejilla de configuraciones.

Este experimento es una modificación del experimento presentado en [9]. También se puede consultar el experimento en el siguiente enlace: https://doc.freefem.org/tutorials/navierStokesProjection.html

Se ha modificado la constante de viscosidad cinemática ν en este experimento respecto de la original a 0.00025 para que el fluido sea mucho más turbulento (y por tanto incremente la complejidad de la función que tiene que construir la PINN). El experimento en cuestión simula el comportamiento de un fluido en dos dimensiones en el interior de un tubo que se ensancha cerca de su lado izquierdo. El tubo tiene una entrada en dicho lado, y una salida en el lado derecho. Los lados superior e inferior están cerrados. En la Figura 3.6 se muestran el mallado del problema y las predicciones del modelo numérico al final del experimento.



(a) Mallado del problema construido en FreeFEM



(b) Predicción de FreeFEM en t=30

Figura 3.6: Mallado y predicción del modelo numérico para el problema 4.

A través del lado izquierdo hay una entrada de fluido dada por la función $f(y) = 4 \cdot y \cdot (1 - y)$, que dura desde el inicio del experimento en $t = 0$ hasta su final en $t = 30$. A medida que avanza el tiempo del experimento, se generan turbulencias en los laterales del tubo. El sistema de ecuaciones que rige el

experimento es:

$$\begin{cases} \partial_t u + u \cdot u_x + v \cdot u_y - u_{xx} - u_{yy} + p_x = 0 & \text{Sobre } \Omega & \text{(3.5a)} \\ \partial_t v + u \cdot v_x + v \cdot v_y - v_{xx} - v_{yy} + p_y = 0 & \text{Sobre } \Omega & \text{(3.5b)} \\ u_x + v_y = 0 & \text{Sobre } \Omega & \text{(3.5c)} \\ u, v, p = 0 & \text{En } \{t = 0\} & \text{(3.5d)} \\ u = 4 \cdot y \cdot (1 - y), v = 0 & \text{En } \Gamma_1 & \text{(3.5e)} \\ u\big|_{\Gamma_2} = u_{\Gamma_2}, v\big|_{\Gamma_2} = v_{\Gamma_2} & \text{En } \Gamma_2 & \text{(3.5f)} \\ p = 0, v_{\Gamma_3} = 0 & \text{En } \Gamma_3 & \text{(3.5g)} \end{cases}$$

Donde u es la velocidad horizontal del fluido, v es su velocidad vertical, y p la presión en un punto dado. Γ_1 es el borde izquierdo que hace de entrada de fluido, Γ_2 son los bordes del tubo, y Γ_3 es la salida de fluido en el extremo derecho.

El efecto que se debería observar en las predicciones es un conjunto de turbulencias en los bordes del tubo producidas por las propiedades del fluido y la forma del tubo, y un flujo principal que va desde la entrada hasta la salida, el cual es desplazado ligeramente por las turbulencias.

Con esta sección quedan explicados todos los problemas que componen la experimentación. Pasemos a hablar de la rejilla de hiperparámetros sobre la que se realizará la búsqueda de las mejores configuraciones a lo largo de la experimentación en los 3 primeros problemas.

3.4.5. Rejilla de hiperparámetros.

A la hora de evaluar el alcance de una PINN no resulta suficiente con montar una única red con unos parámetros fijos y estudiar su comportamiento. Esto se debe a que la capacidad de una red neuronal de aproximar una función objetivo esta fuertemente condicionada por los parámetros elegidos a la hora de ajustarla. Por tanto, en cada experimento se ha construido una rejilla de parámetros que consiste en algunas combinaciones posibles de los 7 parámetros sobre los que hemos realizado variaciones. Los parámetros son los siguientes:

- Densidad de puntos: Número de puntos por unidad de área o volumen del espacio simulado. Dado que cada experimento tiene distintas dimensiones también se ha registrado el número total de puntos.

- Función de activación: Existen numerosas funciones de activación para las redes neuronales en el paquete que Sciann usa de backend, Keras. Por ello sólo se ha experimentado con algunas de las más comunes.
- Loss: Sciann sólo permite dos funciones de pérdida, el MAE y el MSE. Ambas se han considerado como parámetros en todas las experimentaciones.
- Optimizador: El optimizador actúa sobre la mejora de pesos en el proceso del descenso de gradiente. Se ha evaluado el rendimiento de unos pocos de los optimizadores disponibles en Keras.
- Batch size: El batch size influye en la velocidad de computación y en el número de pasos necesarios para alcanzar buenos resultados. Se ha experimentado con distintos valores en función del tamaño del experimento.
- Épocas de entrenamiento: El número de épocas determina en gran medida cómo de bien absorbe una red el conjunto de datos de entrenamiento. Se ha variado este parámetro atendiendo también al tamaño del experimento.
- Estructura de la red neuronal: Todas las redes neuronales de la experimentación son Redes Neuronales Densas (DNN), con distintos números de capas y neuronas.

Dado que la rejilla de posibles combinaciones de parámetros es demasiado extensa, en todos los casos se ha dividido la experimentación en dos fases. En la primera se experimenta sobre los parámetros densidad de puntos, función de activación, loss y optimizador, fijando el valor del resto de parámetros. En la segunda fase se experimenta sobre el número de épocas, el batch size y la estructura de la red neuronal.

3.5. Métricas de evaluación.

Por último es necesario evaluar el rendimiento de cada modelo para conocer tanto el rendimiento general de la técnica de modelado mediante PINN, como el rendimiento específico de cada modelo en función de los ajustes introducidos: arquitectura de la DNN, función de activación, densidad del mallado de entrenamiento, etcétera. Para hacerlo se ha utilizado el paquete MLflow, que permite registrar parámetros, resultados y gráficas de cada ejecución del entrenamiento del modelo. Para hacerlo, en cada experimento se ha implementado un bucle

que entrena y evalúa el modelo con los ajustes que se le pasen como parámetros, y que registra toda la información pertinente de cada ejecución.

Para evaluar cada modelo se ha dispuesto un mallado de test ajustado a cada problema, sobre el que se calculan los valores del modelo de validación que estemos utilizando (el modelo simbólico o el numérico). Después se evalúa el rendimiento de la PINN comparando sus resultados con el modelo de validación sobre el mismo mallado, utilizando distintas métricas.

3.5.1. Métricas propuestas para la evaluación.

A la hora de evaluar cada modelo sobre todo nos interesa obtener información sobre las siguientes características de este: exactitud de la solución, medida en que cumple la ecuación diferencial (o el sistema de ecuaciones diferenciales) que rige el problema, el máximo error absoluto cometido, y los tiempos de entrenamiento y evaluación. Como ya hemos dicho, en todos los casos, estas medidas se tomarán sobre un mallado de test fijo, que se construirá antes de iniciar cada bucle de experimentación.

Para evaluar la exactitud de la solución vamos a acudir a las normas L^p, en concreto a las normas L^1 y L^2. Estas dos normas definen la distancia entre dos funciones en un espacio métrico (un espacio vectorial sobre el que se puede integrar). Así, si tenemos dos funciones f y g sobre el espacio métrico Ω, sus distancias serían:

$$d_1(f,g) = \int_\Omega |f(x) - g(x)|dx \qquad \text{Con la norma } L^1 \tag{3.6}$$

$$d_2(f,g) = \sqrt{\int_\Omega (f(x) - g(x))^2 dx} \qquad \text{Con la norma } L^2 \tag{3.7}$$

Dado que la salida del modelo generado por la PINN es a efectos una función (aunque solo podamos evaluarla en una cantidad finita de puntos), podemos considerar utilizar estas medidas para calcular la distancia entre la salida de la PINN y nuestra función de validación (sea esta obtenida mediante un método numérico o uno simbólico). No obstante, dado que la salida de la PINN no se puede integrar, se ha considerado una aproximación de dichas normas. En lugar de calcular la integral, se puede calcular $|f(x) - g(x)|$ o $(f(x) - g(x))^2$ respectivamente, y ponderar el resultado por la densidad local del punto x. Esta densidad debería calcularse de manera que la suma de densidades equivalga al volumen total del espacio sobre el que se realiza el experimento. Dado que todos

los mallados de test que se han utilizado en la experimentación son uniformes, el cálculo se puede simplificar en gran medida, de manera que:

$$d_1(f,g) = \int_\Omega |f(x) - g(x)|dx \approx MAE(f,g) \cdot \int_\Omega dx \tag{3.8}$$

$$d_2(f,g) = \sqrt{\int_\Omega (f(x) - g(x))^2 dx} \approx RMSE(f,g) \cdot \int_\Omega dx \tag{3.9}$$

Con estas dos fórmulas tendríamos una medida razonablemente robusta para evaluar cómo de parecida es la función obtenida por la PINN a la función de validación obtenida por el método clásico. En este caso, sólo habría que definir f como las salidas de la PINN, y g como las salidas del modelo de validación en el mallado de test.

Para evaluar el Error Máximo simplemente es necesario calcular las predicciones de la PINN y las predicciones del método de validación sobre el mallado de test. Estas predicciones se obtienen como arrays de numpy. Después se calcula el valor absoluto de la diferencia de dichos arrays, y se extrae el valor máximo del array resultante.

A la hora de evaluar una PINN, nos interesa conocer en qué medida cumple la ecuación diferencial que la rige, es decir, en qué medida ha conseguido resolver dicha ecuación. Para hacer esto sería necesario calcular diferenciales sobre la PINN, para poder obtener una función como las funciones H_i definidas en (2.32). Por suerte, el paquete SciAnn genera internamente una serie de salidas sobre la PINN que calculan esta clase de funciones H_i. Por tanto, para evaluar el grado de cumplimiento de una ecuación diferencial, solo es necesario localizar esta salida, evaluarla sobre todos los puntos del mallado de test, y calcular el MAE sobre el conjunto de valores resultante. Esta métrica, a la que llamaremos Funcional, se ha empleado únicamente en los primeros tres problemas, pues en el último sería necesario añadir modificaciones a su definición, y no aportará mucho. En los casos en los que se ha usado, todos ellos regidos por la ecuación del calor, su fórmula es la siguiente:

$$\text{MAE}\left(\frac{\partial}{\partial t}\mathcal{N}(W; X^*, T^*) - \Delta\mathcal{N}(W; X^*, T^*)\right), \tag{3.10}$$

donde $\mathcal{N}(W; X^*, T^*)$ es la red neuronal que compone la PINN $\mathcal{N}$ con los pesos W, evaluada sobre el mallado (X^*, T^*). En definitiva, la proximidad a 0 de esta métrica nos explica con qué precisión cumple la PINN la ecuación del calor.

Por último, para evaluar el tiempo de entrenamiento y evaluación simplemente se han medido los tiempos de ejecución en el equipo. Como ya se explicó en la Sección 3.1, con el objetivo de que la comparativa de tiempos entre modelos tenga sentido, se ha utilizado el mismo equipo para realizar todas las ejecuciones.

Capítulo 4

Experimentación.

En este capítulo se comentarán los resultados obtenidos en cada uno de los experimentos propuestos en la Sección 3.4. El objetivo de estos experimentos es evaluar cómo de precisa es una PINN a la hora de simular un fenómeno físico.

Por simplicidad a la hora de exponer y analizar resultados, todos los experimentos sobre los que se ha realizado una búsqueda de parámetros consisten en simulaciones del comportamiento del calor en dimensiones crecientes. De este modo, el primer experimento simula el calor sobre una barra (1 dimensión), el segundo sobre una placa (2 dimensiones), y el tercero sobre un cubo (3 dimensiones). El último experimento es una simulación de un fluido turbulento en 2 dimensiones.

Como ya explicamos en el Capítulo 3, todos los experimentos se ha realizado una búsqueda sobre 7 parámetros de generación del modelo:

- Densidad de puntos.
- Función de activación.
- Loss.
- Optimizador.
- Batch size.
- Épocas de entrenamiento.
- Estructura de la red neuronal.

También cabe recordar que dado que la lista de posibles combinaciones de los anteriores parámetros es demasiado amplia para realizar una búsqueda de parámetros sobre ella, todos los experimentos de búsqueda de parámetros se dividen en dos fases. En la primera se experimenta sobre los parámetros densidad de puntos, función de activación, loss y optimizador, fijando el valor del resto de parámetros. En la segunda fase se experimenta sobre el número de épocas, el batch size y la estructura de la red neuronal.

Terminado este pequeño resumen del marco de experimentación, vamos a pasar a comentar los resultados del primer experimento.

4.1. Primer Experimento: Calor sobre una barra.

En esta sección comentaremos los resultados obtenidos en la búsqueda de parámetros realizada sobre el experimento descrito en Subsección 3.4.1. En primer lugar explicaremos de qué modo se ha realizado la experimentación.

4.1.1. Entorno del experimento.

Para realizar la experimentación de esta parte se ha creado un cuaderno JupyterNotebook que realiza toda la búsqueda sobre el mallado de posibilidades. El cuaderno está disponible en [3].

Para la primera fase se han utilizado los siguientes valores:

- Densidades de puntos: 5, 10, 25 y 50.
- Funciones de activación: Tangente Hiperbólica, ReLU, Softmax y Sigmoide.
- Loss: mse, mae.
- Optimizer: SGD, RMSprop, Adam, Nadam y Ftrl.

Se han considerado todas las combinaciones posibles de los anteriores parámetros fijando los tres parámetros restantes en:

- Estructura de la red neuronal: 5 capas con 5 neuronas cada una.
- Épocas de entrenamiento: 1000 épocas.
- Batch size: 20000 muestras por batch. En caso de que el número de muestras sea menor, hay un único batch que contiene todas las muestras.

Para la segunda fase del experimento se han fijado los cuatro parámetros anteriores en aquellos que dieron mejores resultados (en la Sección 4.1.2 se discute qué combinación de parámetros es la mejor a la vista de los resultados).

El mallado de configuraciones para la segunda fase se ha construido combinando los siguientes parámetros:

- Estructura de la red neuronal: Anotando la estructura como "Número de capas"*["Número de neuronas por capa"] las estructuras consideradas son: 3*[100], 5*[50], 50*[10], 100*[5], 5*[5], 20*[20] y 50*[50]. Con las dos primeras estructuras se intenta testear modelos con pocas capas de muchas neuronas , las dos siguientes testean modelos con muchas capas de pocas neuronas. Con los tres últimos se intenta evaluar el rendimiento según la variación de tamaños (de redes muy pequeñas a redes de mayor tamaño).

- Épocas de entrenamiento: Dado que la potencia del equipo utilizado es limitada, se ha optado por limitar el número de épocas a 2500. Los parámetros considerados son: 200, 1000 y 2500 épocas.

- Batch size: 5000, 10000 y 25000.

4.1.2. Análisis de resultados.

Fase 1 del experimento.

En primer lugar nos centraremos en los resultados de la fase 1. Nos interesa principalmente identificar la tupla de ajustes que den buenos resultados en las métricas L1 y L2, y que a la vez no tengan un error máximo muy alto. A ser posible, también nos interesaría seleccionar una configuración de las que cumpla las condiciones anteriores cuyo tiempo de entrenamiento sea bajo. Dado que hay 160 combinaciones posibles, para decidir las configuraciones con estas características, vamos a revisar los 6 mejores resultados ordenados respecto de L1 (Tabla 4.1) y Error Máximo (Tabla 4.2).

Loss	mse	**mse**	mse	mse	mse	mse
Optimizer	Adam	**Adam**	SGD	RMSprop	RMSprop	Nadam
Point_Density	25	**50**	50	10	5	50
u_Activator	tanh	**tanh**	tanh	tanh	tanh	tanh
Funcional	0.170	**0.040**	0.224	0.090	0.101	0.091
L1	2.6	**2.8**	3.2	3.4	3.7	3.7
L2	3.5	**3.6**	5.9	3.9	4.3	4.4
MaxError	1.47	**0.42**	1.85	0.41	0.68	0.49

Cuadro 4.1: Mejores resultados respecto de L1

Loss	mse	**mse**	mse	mse	mse	mse
Optimizer	RMSprop	**Adam**	Nadam	RMSprop	RMSprop	RMSprop
Point_Density	10	**50**	50	50	5	25
u_Activator	tanh	**tanh**	tanh	tanh	tanh	tanh
Funcional	0.090	**0.040**	0.091	0.022	0.101	0.055
L1	3.4	**2.8**	3.7	3.8	3.7	7.0
L2	3.9	**3.6**	4.4	4.4	4.3	8.1
MaxError	0.41	**0.42**	0.49	0.54	0.68	0.82

Cuadro 4.2: Mejores resultados respecto de Error máximo

En este caso, parece que la configuración que mejores resultados da es {mse, Adam, 50, tanh}, pues equilibra buenos resultados en ambas tablas. Por tanto, utilizaremos esta configuración como fija en la fase 2 de este experimento.

Si nos fijamos en ambas tablas, parece ser que todas las configuraciones con mejores resultados tienen los parámetros Loss = "mse" y Función de activación = Tangente Hiperbólica. Vamos a analizar el conjunto completo de soluciones para ver qué ajustes tienden a producir mejores resultados, cuáles tienden a producir peores, y en general detectar comportamientos inusuales.

En primer lugar, al revisar el conjunto de datos se observa un comportamiento bastante extraño en la distribución de los datos al graficar el Error Máximo respecto del error L1. En la Figura 4.1 se puede apreciar claramente una correlación lineal positiva en un subconjunto de los datos, y una negativa en otro subconjunto de los datos, ambas formando una cuña en el punto $(L1 \approx 45, \text{MaxError} \approx 2.5)$

Figura 4.1: Gráfico de resultados en L1 y MaxError

Si nos fijamos en la Figura 4.2 podemos observar que los puntos que se encuentran en esta región de comportamiento extraño son aquellos con un error Funcional de menor grado, muy próximo a 0. Si analizamos los gráficos de las predicciones realizadas por algunos de los modelos en el subconjunto $[0, 0.001]$ vemos que son aproximadamente planos paralelos al plano (X, T), es decir, predicen temperatura constante para todos los puntos en todos los momentos. Si tenemos en cuenta la ecuación del calor, la solución de la temperatura constante la cumple, de ahí que estas predicciones obtengan tan buenas puntuaciones en esta métrica.

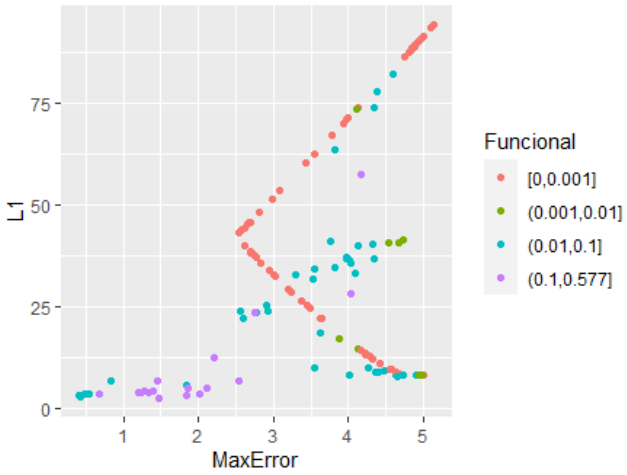


Figura 4.2: Gráfico de resultados en L1 y MaxError, coloreando por segmentos de la métrica Funcional.

Consideremos ahora a estas configuraciones como no viables. Resultaría interesante conocer qué variables tienden a aparecer más en las configuraciones no viables.

Point_Density	n (Max = 40)
5	23
10	25
25	24
50	19

Loss	n (Max = 80)
mae	54
mse	37

Optimizer	n (Max = 32)
Adam	16
Ftrl	26
Nadam	16
RMSprop	14
SGD	19

u_Activator	n (Max = 40)
relu	11
sigmoid	37
softmax	38
tanh	5

Cuadro 4.3: Cantidad de configuraciones no viables en función de cada parámetro.

En la Tabla 4.3 se puede observar que la densidad de puntos no influye mucho en la no viabilidad de la configuración, aunque con densidad 50 los resultados mejoran. De entre los optimizadores, parece que *Ftrl* tiende a obtener resultados

algo peores, pero el resto tienen una distribución uniforme. En cuanto a la función de pérdida, parece que mae suele intervenir en más configuraciones no viables. Por último, llaman la atención los resultados de las funciones de activación Sigmoide y Softmax, que intervienen en las configuraciones no viables en la mayoría de los casos.

Por tanto podemos asumir que se pueden descartar estas dos funciones de activación de aquí en adelante, pues sus resultados dejan mucho que desear.

Al revisar el gráfico de la Figura 4.3, esta vez retirando las configuraciones no viables, y coloreando por función de activación, se puede observar que la función de activación Tangente Hiperbólica interviene en la mayoría de configuraciones con mejores resultados. Las distribuciones resaltando el resto de variables parecen mostrar una distribución con tendencias menos claras, aparte de las comentadas anteriormente.

Figura 4.3: Gráfico de resultados en L1 y MaxError, retirando configuraciones no viables y coloreando por función de activación.

En resumen, de esta primera fase hemos decidido que la configuración fija de la siguiente fase será (Adam, 50, tanh, mse), que la función de activación Tangente Hiperbólica da los mejores resultados, mientras que las funciones de activación Sigmoide y Softmax no son útiles, y que el uso de mae como función de pérdida o *Ftrl* como optimizador empeora los resultados.

Fase 2 del experimento.

En este caso los resultados obtenidos han estado mucho más afinados que en la anterior fase en cuanto al Error Máximo, al L1 y al L2. Primero vamos a echarle un vistazo a los resultados sobre L1, Error Máximo y Funcional mostrados en la Figura 4.4.

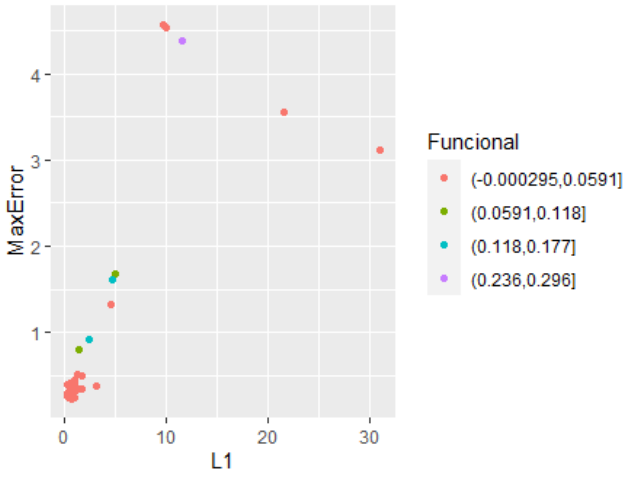


Figura 4.4: Resultados de la fase 2 del experimento 1.

Parece que hay un subconjunto con malos resultados formando una línea del mismo estilo que la que vimos en la fase anterior. Además se observa una gran cantidad de soluciones con una puntuación aproximadamente igual en las tres métricas representadas. Si prestamos atención al valor de Funcional de cada uno de los resultados vemos que salvo una, todas las predicciones tienen un valor de funcional muy bajo.

Por tanto tenemos de nuevo un subconjunto de soluciones que podemos considerar no viables. Se ha analizado este subconjunto para determinar si existe algún parámetro que dé lugar a esta clase de resultados con mayor frecuencia, y se ha determinado que existen ciertas estructuras de red neuronal de entre las propuestas que dan lugar a esta clase de resultados. En la Tabla 4.4 se muestra el número de soluciones no viables por cada estructura propuesta.

Como se puede observar, parece que las redes con muchas capas de pocas neuronas son las que peores resultados dan como conjunto. Por otro lado parece que la estructura 50[50] también ha dado malos resultados por ser una red demasiado grande como para poder ser entrenada. El único resultado de esta

u_Structure	n (Max =9)
100[5]	9
20[20]	1
5[5]	1
50[10]	9
50[50]	8
3[100]	0
5[50]	0

Cuadro 4.4: Número de soluciones no viables por estructura de la PINN

última estructura que ha dado lugar a una solución viable ha sido el que usa 200 épocas de entrenamiento y batch size de 5000. Es posible que esta anomalía se deba a que los pesos iniciales de la red neuronal estaban cerca de componer una buena solución, aunque no podemos estar seguros de ello con los datos disponibles. Por otro lado, las dos soluciones no viables con estructuras 20[20] y 5[5] son aquellas que utilizan 200 épocas de entrenamiento, por lo que no parece que estos parámetros condicionen la mala calidad de la solución. Como conclusión, a partir de este punto excluiremos del análisis a todas aquellas configuraciones que tengan una estructura 100[5], 50[10] o 50[50].

Vamos continuar discriminando los resultados por estructura de la red neuronal. En la Figura 4.5 se puede ver que las estructuras que mejores resultados dan son las caracterizadas por tener pocas capas de gran cantidad de neuronas. Si tenemos en cuenta que todas las estructuras que hemos descartado hasta ahora tenían gran cantidad de capas, parece lógico pensar que, al menos para este problema de una única dimensión, las redes de este tipo convergen mejor hacia la solución.

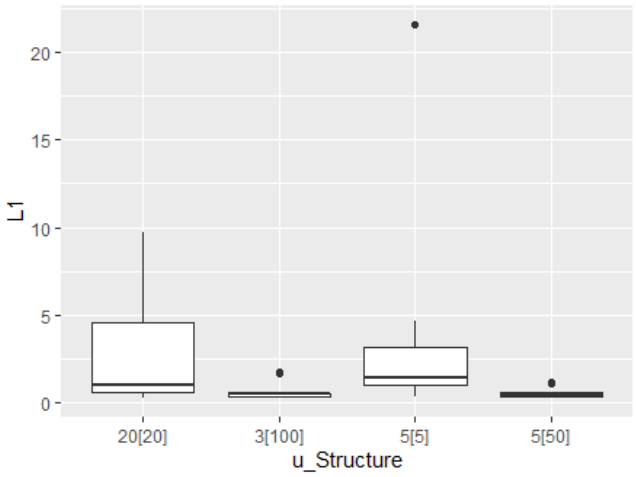


Figura 4.5: Representación de las puntuaciones en L1 por estructura de la red neuronal.

En la Figura 4.6 se han agrupado los dos tipos de configuraciones en "Flat" (aquellas configuraciones con pocas capas y muchas neuronas) y "Square"(aquellas configuraciones con el mismo número de capas y de neuronas por capa) y se ha hecho un gráfico de cajas coloreando por número de épocas de entrenamiento. Como se puede observar, no sólo los resultados son mucho mejores en general utilizando una configuración "Flat" que una configuración "Square", sino que estas primeras configuraciones convergen mucho más rápido hacia el resultado, pues en 200 épocas ya se alcanzan cotas muy bajas de error L1. El outlier de Square que tiene una puntuación muy alta en L1 es una PINN con estructura 5[5] que también aparece en la Figura 4.5. Teniendo en cuenta ambos gráficos, dado que la velocidad de entrenamiento para las estructuras Square parece mejorar a medida que su tamaño se reduce, esta anomalía podría deberse a los pesos iniciales de la PINN.

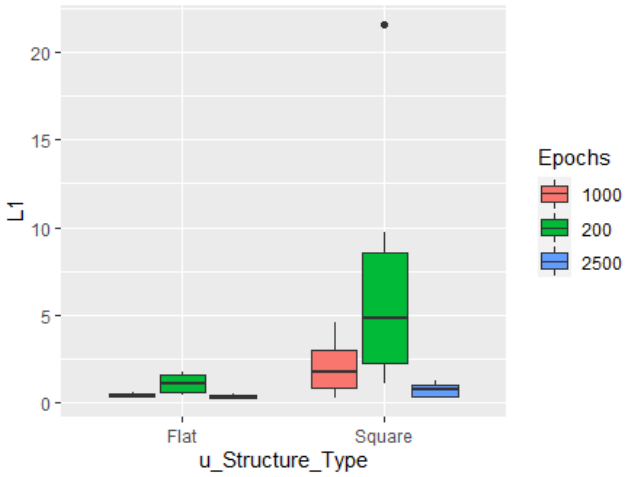
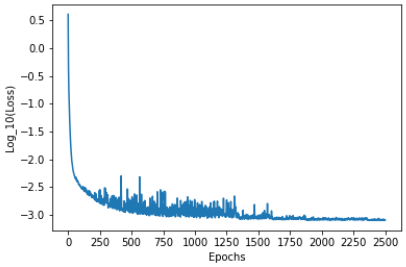
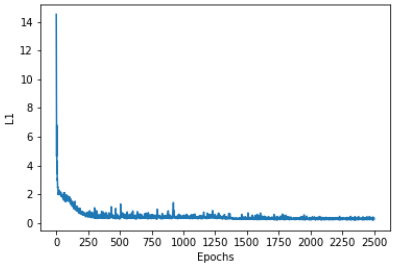


Figura 4.6: Puntuaciones en L1 en función del tipo de estructura y épocas de entrenamiento

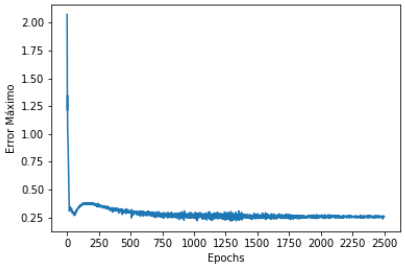
Por último resta analizar la mejor solución alcanzada y compararla con la solución dada por el método numérico. Para ello se ha seleccionado la configuración (Batch size = 500, Epochs = 2500, Estructura = 3[100]), y se ha repetido el experimento, esta vez registrando las métricas L1 y Error Máximo en cada iteración. El resultado se muestra en la Figura 4.7. Como se puede observar, el valor de Loss decrece muy rápido, alcanzando el orden de 10^{-3} en 250 épocas. En esta cantidad de iteraciones, el valor de L1 ya está en torno a 0, y el Error Máximo ya alcanza cotas muy bajas, aunque fluctúa un poco.



(a) Evolución de Loss



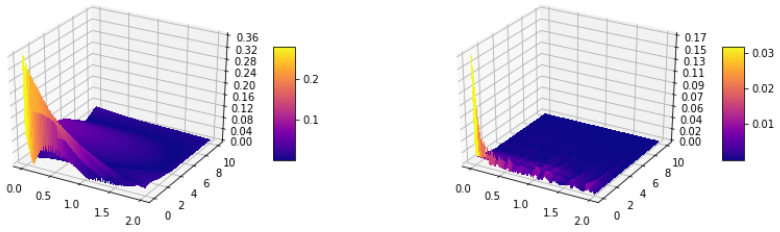
(b) Evolución de L1



(c) Evolución del Error Máximo

Figura 4.7: Evolución de varias métricas a lo largo de 2500 épocas para un modelo con estructura 3[100].

En la Figura 4.8 se muestran los errores respecto del modelo simbólico cometidos por el modelo PINN que dio lugar a la mejor solución y un modelo numérico construido para este problema. Como se puede observar, el modelo numérico no sólo es mucho más preciso, sino que el error que comete es mucho más localizado. Por esta razón, en los siguientes experimentos se han empleado modelos numéricos como herramienta de obtención de datos de validación.



(a) Error del modelo PINN (b) Error del modelo numérico

Figura 4.8: Comparativa de los errores del mejor modelo PINN y el modelo numérico.

4.2. Segundo Experimento: Foco de calor en una placa.

4.2.1. Entorno del experimento.

Para realizar la experimentación de esta parte se ha generado otro cuaderno JupyterNotebook que realiza toda la búsqueda sobre el mallado de posibilidades. El cuaderno está disponible en [3].

Para este experimento se ha reducido un poco el tamaño del espacio de búsqueda, pues es computacionalmente más exigente que el anterior. Para ello se han tenido en cuenta los parámetros que dieron malos resultados en la anterior experimentación y se han retirado de la malla de parámetros, para ser sustituidos por otros. También se ha reducido el tamaño de las densidades de puntos debido al mayor tamaño del espacio experimental.

En la primera fase se han utilizado los siguientes valores:

- Densidades de puntos: 2, 3 y 4.
- Funciones de activación: Tangente Hiperbólica, ReLU, SoftPlus.
- Loss: mse, mae.
- Optimizer: SGD, Adam y RMSprop.

De entre los tres parámetros restantes, la estructura de la red neuronal se ha elegido atendiendo a la mejor candidata en la experimentación anterior. Los

otros dos parámetros se han tomado con el objetivo de limitar la duración de la experimentación:

- Estructura de la red neuronal: 3 capas con 100 neuronas cada una.
- Épocas de entrenamiento: 2000 épocas.
- Batch size: 20000 muestras por batch. En caso de que el número de muestras sea menor, hay un único batch que contiene todas las muestras.

En la fase 2 se ha utilizado la mejor combinación de parámetros resultante de la fase 1, y se ha variado sobre la malla formada por los siguientes parámetros:

- Estructura de la red neuronal: 3*[100], 20*[20], 4*[200], 30*[30] y 5*[50]
- Épocas de entrenamiento: 100, 200, 500 y 1000 épocas.
- Batch size: 5000, 10000, 20000. Hay que tener en cuenta que contamos con un total de 20000 muestras, por lo que en el último caso hay un único bloque de datos por época.

4.2.2. Análisis de resultados.

Fase 1 del Experimento 2.

En el caso de este experimento, los resultados en cuanto a la función de activación que mejores puntuaciones consigue han sido muy claros. En la Figura 4.9 se puede comprobar fácilmente que la función Tangente Hiperbólica es la que mejor se ajusta a la función objetivo, seguida de la función ELU, y que la función ReLU rinde bastante peor que las dos anteriores. Además, parece que en general las puntuaciones en Error Máximo han empeorado de manera general en este experimento. Comentaremos este problema más adelante.

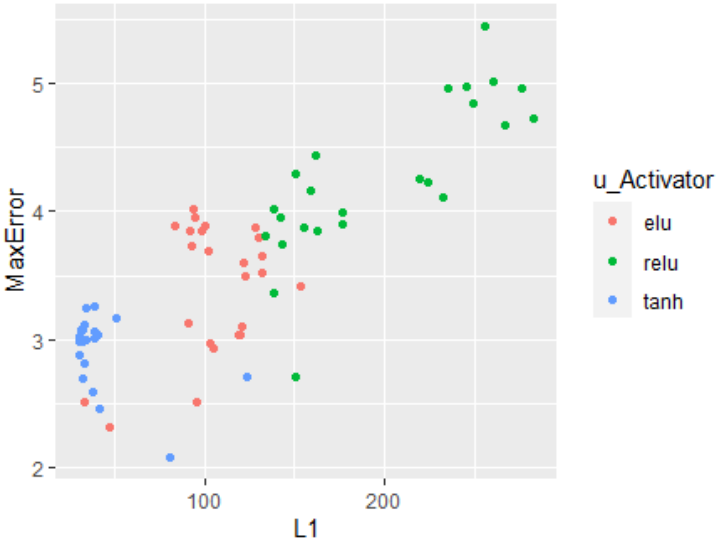
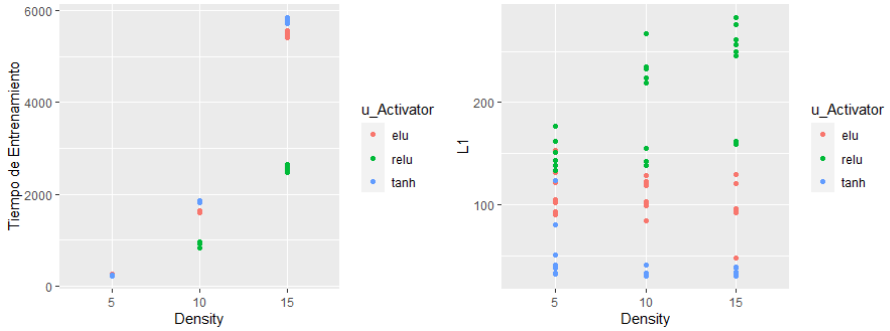


Figura 4.9: Resultados en L1 y MaxError, coloreando por función de activación.

En cuanto al resto de parámetros, parece que tanto el Loss como el Optimizador elegido no influyen mucho en el resultado final. No obstante, sí que hay un punto interesante en cuanto al parámetro Densidad de puntos, pues mientras que este parámetro influye mucho en el tiempo de computación necesario para entrenar una PINN, en este caso no parece aportar mucho a la mejora de la red neuronal. En la Tabla 4.5 se muestra la cantidad de puntos que contiene el mallado de entrenamiento en relación al parámetro Densidad escogido, y en la Figura 4.10 se muestran, por un lado el tiempo de entrenamiento consumido en relación a la Densidad y de la Función de Activación, y por otro la puntuación en L1 alcanzada, en relación también a estos dos parámetros. Parece claro que no merece la pena considerar un valor de densidad de puntos superior a 10, dado que el gasto en tiempo de computación no merece la pena si tenemos en cuenta los resultados alcanzados.

Densidad	Num Puntos
15	540.000
10	160.000
5	20.000

Cuadro 4.5: Relación del Número de puntos en función del valor de Densidad escogido



(a) Tiempo de entrenamiento empleado en función del parámetro Density

(b) Puntuación en L1 en función del parámetro Density

Figura 4.10: Evolución de varias métricas a lo largo de 2500 épocas para un modelo con estructura 3[100]

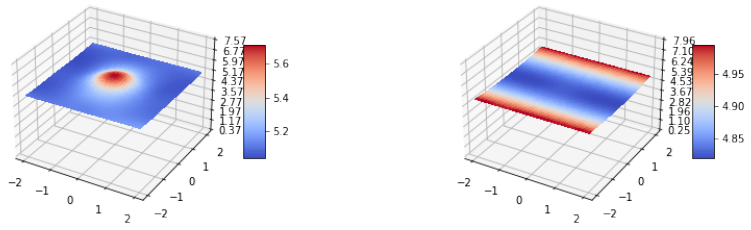
Teniendo en cuenta estos resultados, vamos a considerar como la mejor configuración a aquella que haya obtenido mejor puntuación y tenga como función de activación la Tangente Hiperbólica y una cantidad baja de puntos de entrenamiento. Dado que incluso con una Densidad de 10 puntos, el tiempo de computación medio para entrenar una PINN supera los 30 minutos, se ha decidido que la Densidad de puntos que se va a utilizar en la siguiente fase será de 5. Revisando los mejores resultados respecto de L1 y Error Máximo se ha determinado que la mejor configuración para la siguiente fase es (Adam, 5, tanh, mae).

Fase 2 del Experimento 2.

Antes de pasar a analizar los resultados de esta parte, conviene comentar un comportamiento anómalo que se ha detectado en buena parte de las simulaciones construidas. Como ya hemos comentado, los datos que estamos utilizando para la validación provienen de una simulación y no de un experimento real, por lo que pueden no adaptarse a la realidad. Como se muestra en el foco de calor se enciende en $t = 5$ y va aumentando su temperatura hasta estar completamente encendido. Este foco ha sido definido atendiendo a la facilidad de implementación de usar una función gaussiana para generarlo, y no a su verosimilitud con un foco de calor real, por lo que es posible que no simule este fenómeno correctamente.

Por esta razón, al analizar una simulación generada mediante una PINN, y

compararla con la simulación obtenida de FreeFEM, se pueden observar algunas diferencias en la forma del foco en cuanto a las coordenadas espaciales, pero también en las temporales. Esto quiere decir que para equilibrar la aparición repentina de un foco de calor en $t = 5$, en ocasiones la PINN lo "empuja" hacia atrás en el tiempo, haciendo que aparezca este foco antes de dicho momento (ver Figura 4.11).

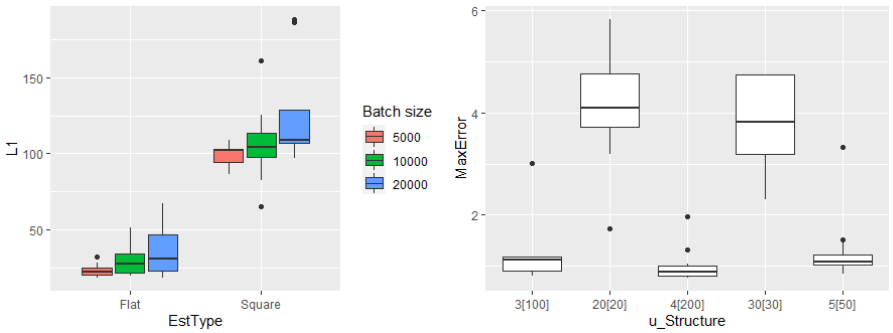


(a) Simulación basada en PINN. (b) Simulación numérica.

Figura 4.11: Comparación de la simulación numérica y la simulación basada en PINN en el experimento 2 en $t = 5n$

Esta clase de diferencias entre simulaciones (que no se daban en el experimento anterior) implican que seguramente ningún modelo alcanzará cotas de L1, L2 y Error Máximo muy bajas como las que vimos en el anterior experimento.

Una vez comentado esto, pasemos a analizar los resultados de esta parte del experimento. En primer lugar, de nuevo las estructuras de red neuronal de pocas capas y muchas neuronas han superado al resto en todas las métricas, como se puede ver en la Figura 4.12.



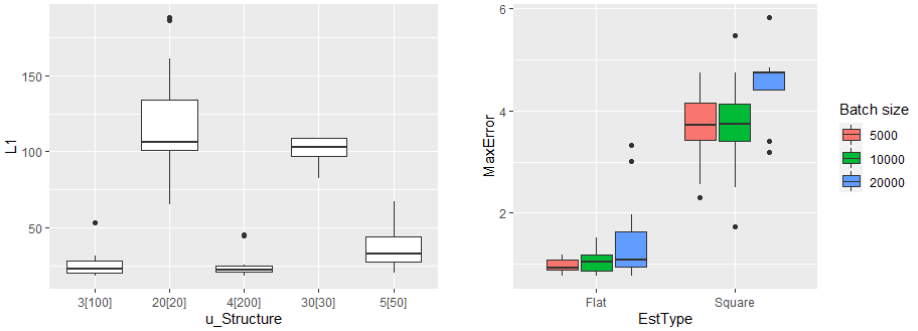
(a) Puntuaciones en L1 según la estructura de la PINN

(b) Puntuaciones en Error Máximo según la estructura de la PINN

Figura 4.12: Evolución de las PINN en función de su estructura

Por lo que hemos visto hasta ahora, podemos asumir que estas estructuras "Flat" son las que mejores puntuaciones alcanzan, independientemente de la dimensión. En el siguiente experimento comprobaremos si este comportamiento se repite con 3 dimensiones.

Además de obtener mejores resultados, esta clase de estructuras convergen mucho mejor hacia una solución suficientemente buena (como cabría esperar, pues al tener menos capas, el algoritmo de backpropagation se ve menos distorsionado). En la Figura 4.13 se muestra cómo las estructuras "Flat" ya alcanzan un buen nivel de precisión en torno a la época 200, y en la época 500 ya alcanzan la precisión máxima posible, pues apenas evolucionan en 500 épocas más. En cuanto al Batch Size, como también vimos en el anterior experimento, la reducción de este suele ir acompañada de mejores puntuaciones en L1 y Error Máximo.



(a) Evolución en L1 en función de la cantidad de épocas de entrenamiento

(b) Evolución en Error Máximo en función de la cantidad de épocas de entrenamiento

Figura 4.13: Evolución de las PINN en función de épocas de entrenamiento

4.3. Tercer Experimento: Pozo de calor en un bloque.

4.3.1. Entorno del experimento.

De nuevo, para realizar la experimentación de esta parte se ha generado otro cuaderno que realiza la búsqueda sobre el mallado de parámetros, que está disponible en [3]

En este caso, la duración del experimento es menor que en los anteriores. Esto se debe a que para construir un mallado sobre este con una densidad de puntos de hasta 5 por unidad de volumen, es necesario alcanzar cantidades de puntos en torno a los 200.000. Dado que el mallado de parámetros sobre el que se pretende experimentar es bastante amplio, se ha limitado esta duración en la medida de lo posible para limitar el tiempo de experimentación. Debido a esta reducción en la densidad de puntos, se ha optado por superponer dos mallados de densidad máxima 4, uno de ellos en forma de cuadrícula como se había venido utilizando hasta ahora, y otro de ellos radial con radio 1 centrado en la región interna del pozo de calor. Con esto se consigue que haya una buena cantidad de muestras de datos en la zona con mayor variabilidad, que es la del pozo de calor.

En esta búsqueda de parámetros se ha reducido aún más el mallado de búsqueda, descartando algunos parámetros que dieron malos resultados en experimentos anteriores.

En la primera fase se han utilizado los siguientes valores:

- Densidades de puntos: 5, 10 y 15.
- Funciones de activación: Tangente Hiperbólica, ReLU, Elu.
- Loss: mse, mae.
- Optimizer: SGD, Adam y Nadam.

Al igual que en las anteriores ocasiones, se han fijado los tres parámetros restantes atendiendo a la mejor combinación en el experimento anterior y a la reducción del tiempo de entrenamiento de cada modelo. En este caso, se ha tomado como estructura de la red neuronal a 3*[100] de nuevo, 1000 épocas de entrenamiento, y Batch Size 2000. Los parámetros fijados para la segunda fase se decidieron en función de los resultados de la fase 1, y se discuten en

En la segunda fase se han utilizado los siguientes valores:

- Estructura de la red neuronal: 3*[200], 4*[100], 5*[100] y 6*[60].
- Épocas de entrenamiento: 1000, 500 y 100 épocas.
- Batch size: 5000, 10000, 15000.

4.3.2. Análisis de resultados.

Fase 1 del experimento 3.

En este caso, el bucle del experimento ha dado algunos resultados anómalos. Parece ser que en algunos casos el entrenamiento de la PINN no solo no ha conseguido converger a una solución aceptable, sino que ha divergido. Esto quiere decir que los pesos de la red neuronal se han ido desajustando progresivamente durante el entrenamiento, para finalmente dar lugar a salidas muy alejadas de cualquier predicción con sentido. Por ejemplo, el error máximo en estos resultados anómalos va desde 120 hasta $8 \cdot 10^{18}$, cuando la variabilidad máxima en la temperatura del experimento es de alrededor de 4 unidades.

Retirando estos resultados anómalos y haciendo una gráfica de L1 vs. Error Máximo, obtenemos la Figura 4.14.

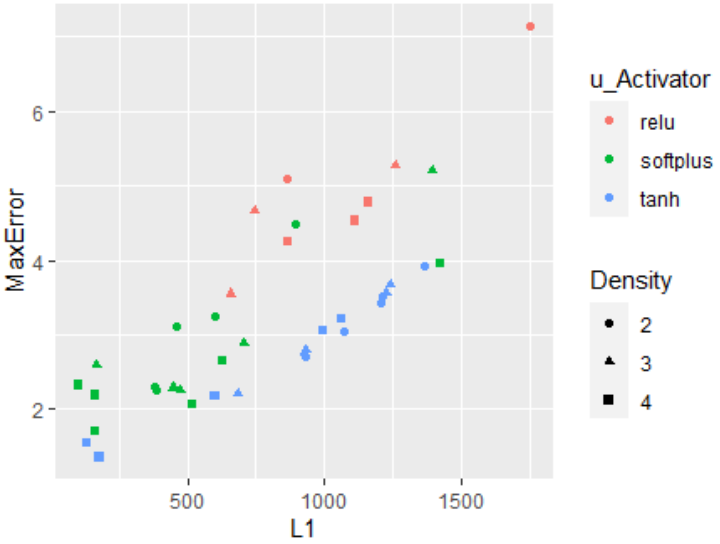


Figura 4.14: Gráfica de resultados del experimento 3.

Como se puede ver, las mejores configuraciones son aquellas con densidad 4, y una densidad menor empeora los resultados. También parece que la función de activación Softplus ha conseguido obtener soluciones con valores en L1 próximos a los de la Tangente Hiperbólica, pero aún así se queda atrás en Error Máximo. Por último, no se observa ninguna tendencia clara en cuanto a los optimizadores, y parece que en este caso el parámetro Loss, MAE lidera el ranking de mejores resultados en todas las métricas.

Si le echamos un vistazo a la Tabla 4.6, vemos que existen varias configuraciones que cometen un error máximo inferior a 2, y mantienen un L1 menor a 200. De entre estas tomaremos la configuración de la segunda fila (Adam, tanh, mae, 4) como configuración para la siguiente fase de este experimento.

L1	MaxError	Optimizer	u_Activator	Loss	Density
101.01	2.32	Adam	softplus	mae	4
133.08	1.54	Adam	tanh	mae	4
163.41	1.70	Adam	softplus	mse	4
163.59	2.19	RMSprop	softplus	mae	4
166.59	2.59	Adam	softplus	mae	3
177.91	1.35	RMSprop	tanh	mae	4

Cuadro 4.6: Mejores resultados respecto de L1 en la fase 1 del experimento 3.

Por otro lado, parece que hay un aumento bastante generalizado del error en L1. Esto podría deberse a dos razones: o bien las posibles densidades de puntos que hemos tenido en cuenta son insuficientes para que la red neuronal realice una buena predicción en base a los datos disponibles, o bien la estructura de la red neuronal son insuficientes para absorber la complejidad de esta solución. En la siguiente fase de la experimentación podremos ver si se trata de esta segunda razón.

Fase 2 del experimento 3.

En esta fase del experimento se han observado comportamientos parecidos a los que vimos en el experimento 2. Las PINN con estructura 3*[200] han obtenido mejores resultados en general, y la reducción del batch size, y el incremento del número de épocas de entrenamiento también consiguen mejorar bastante dichos resultados. A pesar de que el comportamiento es el mismo que en el anterior experimento, vamos a comentar un comportamiento que ha aparecido en todas las PINN que han conseguido buenas puntuaciones en todas las métricas. Si nos fijamos en la Figura 4.15 podemos ver que hay un comportamiento un tanto extraño al inicio de la simulación.

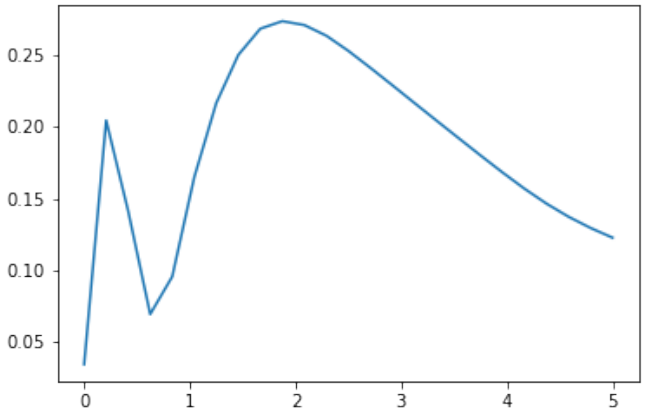


Figura 4.15: Error medio a lo largo del tiempo en un modelo del experimento 3.

Parece ser que cuando existe una discontinuidad muy fuerte en los parámetros que se le pasan a la PINN, esta tiene problemas para adaptarse. Un ejemplo de discontinuidad en los parámetros iniciales sería la de este experimento. En

este caso el espacio del experimento comienza a una temperatura constante, y a partir del momento 0 aparece un foco de calor (de temperatura no constante), lo que implica que hay un cambio muy brusco de temperatura en los datos que recibe el modelo. Mientras que el método numérico no tiene problema en adaptar esta clase de saltos en las condiciones iniciales, la PINN se desajusta momentáneamente, y después se ajusta y comienza a divergir poco a poco respecto del modelo numérico, pero de manera más suave. Este comportamiento es parecido al que observamos en el experimento 2, en el que al encontrarse con un cambio brusco, la PINN empuja hacia atrás en el tiempo el foco de calor para conseguir continuidad.

La principal fuente de este problema es que la PINN considera al tiempo como una dimensión más, que se puede recorrer en ambos sentidos. Por tanto, si existe alguna discontinuidad muy fuerte por la aparición de un foco, o una discontinuidad en las condiciones iniciales, la PINN completa los datos desplazándolos hacia atrás en el tiempo, para así poder cumplir alguna otra de las condiciones en forma de ecuaciones diferenciales que se le han pasado.

Este fenómeno es un problema para la construcción de experimentos en los que no se dispone de datos reales, como es el caso, pues implica que las predicciones que consiga obtener la PINN no serán muy fiables.

4.4. Cuarto Experimento: Flujo turbulento en dos dimensiones.

Dado que el objetivo de este experimento es mostrar las limitaciones actuales de las PINNs, no se ha realizado ningún tipo de búsqueda sobre un espacio de parámetros. Únicamente se han empleado los parámetros que mejores resultados han dado en las anteriores experimentaciones , y se ha entrenado el modelo durante 2500 épocas. El análisis aquí presentado pretende exponer las limitaciones de la técnica utilizada para el entrenamiento de las PINNs.

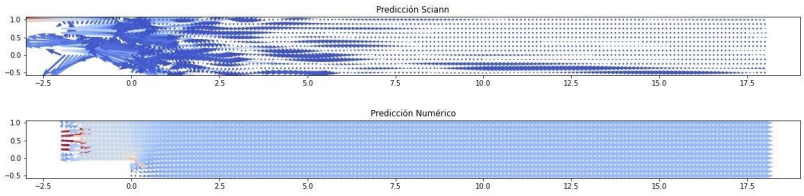
En primer lugar, si hacemos un cálculo de las métricas L1 y Error Máximo en cada una de las magnitudes que intervienen el experimento resulta lo siguiente:

El error L1 en u (velocidad horizontal del fluido) es de 288.897. Una magnitud tan alta, teniendo en cuenta que el valor mínimo obtenido para el modelo numérico es de -0.55 y el máximo es de 1.30 indica que en muchas ocasiones la PINN simplemente predice una dirección contraria a la del modelo numérico. Esto podría indicar que la PINN no consigue predecir las turbulencias presentes en la simulación. El error absoluto máximo en u es de 1.30, por lo que esto

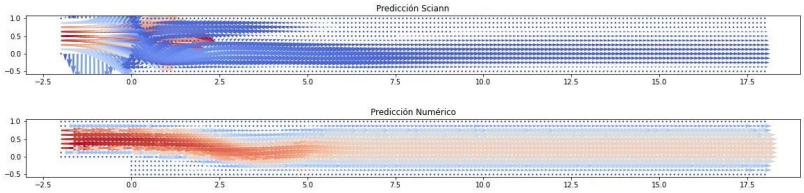
parece muy probable.

Si comprobamos el error L1 en v (velocidad vertical del fluido) vemos que es de 13.617. El rango de valores para v es menor que el de u, desde -0.42 hasta 0.6. El error máximo es de 0.65.

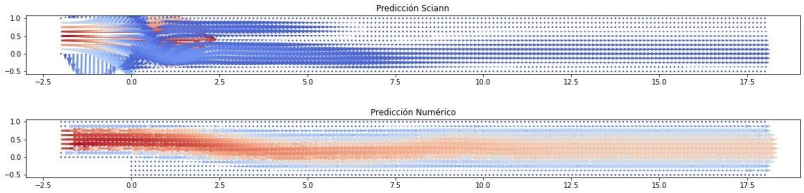
El error en p (presión del fluido) es de 173.764. Este error también es bastante significativo, y parece ser el que más empeora los resultados en la simulación como veremos más adelante.



(a) Representación en t=0



(b) Representación en t=15



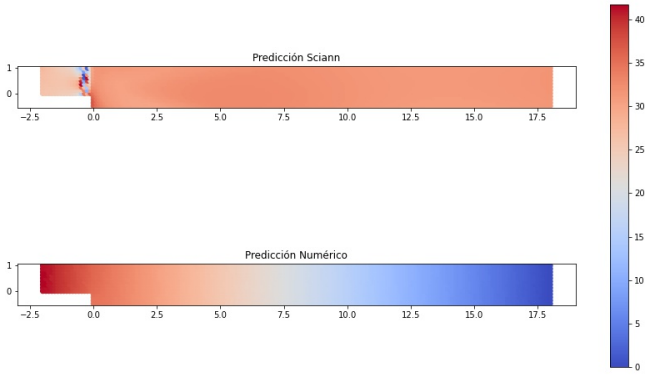
(c) Representación en t=30

Figura 4.16: Gráfico comparativo de las predicciones numéricas y PINN del vector [u,v] en el experimento 4 en 3 momentos distintos.

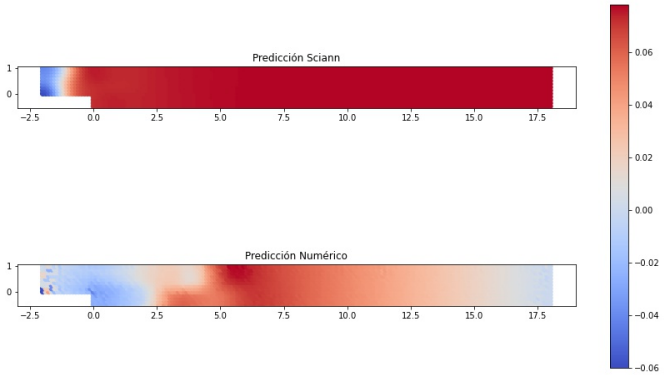
Para poder realizar una comparación gráfica de las predicciones se han generado varias imágenes que comparan las predicciones de la PINN y del método numérico. El primer grupo (Figura 4.16) es un conjunto de tres imágenes que representan el vector velocidad $[u, v]$ en tres momentos distintos mostrando las

predicciones del modelo PINN y el modelo numérico que se ha usado como modelo de validación.

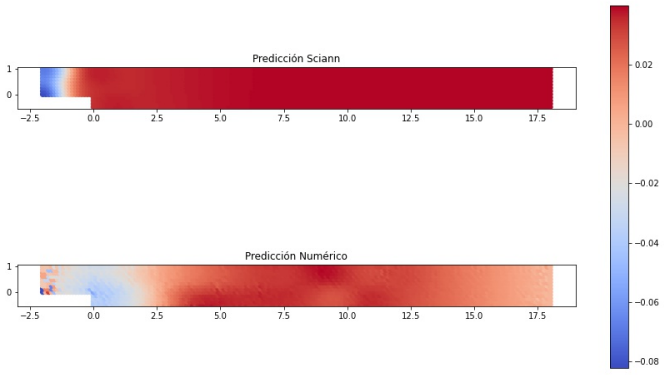
Como se puede observar, las predicciones hechas por el modelo PINN son muy caóticas, pues hay una gran cantidad de turbulencias, y el vector velocidad no parece tener ningún tipo de continuidad. Además, en el lado inferior izquierdo del espacio de simulación los vectores velocidad van cruzan el borde aislado, algo que debería ser imposible dadas las condiciones que le hemos impuesto a la simulación. Esto se debe a otro comportamiento observado en las PINNs: en caso de haber una gran cantidad de funciones que formen parte la función loss, si algunas de ellas se contradicen (es decir, mejorar en una implica empeorar en la otra) entonces la PINN acostumbra a a mejorar únicamente aquella que tiene mas peso o da mejores resultados en vez de buscar equilibrio entre las dos funciones de pérdida.



(a) Representación en t=0



(b) Representación en t=15



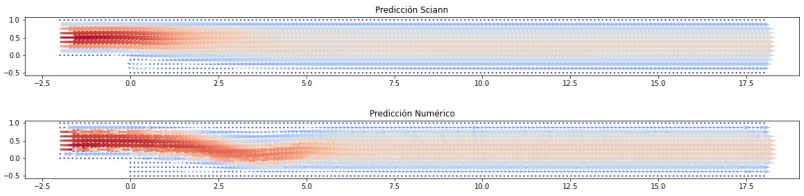
(c) Representación en t=30

Figura 4.17: Gráfico comparativo de las predicciones numéricas y PINN de la presión en el experimento 4 en 3 momentos distintos.

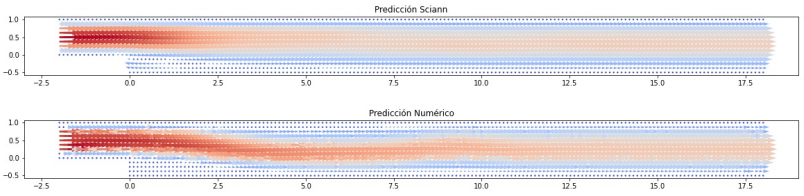
Si revisamos el segundo grupo de imágenes de la simulación, en la Figura 4.17, podemos ver que en general, las predicciones son totalmente distintas. Mientras que las predicciones del modelo numérico muestran irregularidades en la presión dependiendo de la etapa, el modelo PINN tiene unas predicciones bastante uniformes, sobre todo en $t = 15$ y $t = 30$. Esta uniformidad en las presiones es una mala señal, pues precisamente es la diferencia en las presiones la que induce las turbulencias que aparecen en el modelo numérico. Esta diferencia tan fuerte posiblemente provenga de que en este caso el método numérico empleado no ha consistido únicamente en la técnica que habíamos utilizado hasta ahora, sino que incluye un algoritmo llamado algoritmo de Chorin que se utiliza para este tipo de casos específicos de las ecuaciones de Navier-Stokes.

4.4.1. Entrenamiento de la PINN con datos iniciales.

Para finalizar este experimento, se ha modificado el método de entrenamiento de la PINN que habíamos utilizado hasta ahora. Se ha entrenado una nueva PINN, a la cual se le han facilitado, además de la información previa, las predicciones obtenidas por el método numérico desde $t = 0$ hasta $t = 5$. En este caso la PINN ha obtenido resultados algo menos imprecisos.



(a) Representación en t=15



(b) Representación en t=30

Figura 4.18: Comparación de las predicciones del modelo numérico y la PINN con datos del modelo numérico en 2 momentos distintos.

Si revisamos la Figura 4.18, vemos que en este caso las predicciones de la PINN no se diferencian tan fuertemente de las del modelo numérico. En primer lugar no son tan caóticas, y no parecen contradecir las condiciones de contorno que se le impusieron a la PINN. No obstante, sí que presentan un problema muy claro, y es que el modelo PINN no predice ninguna clase de turbulencia. Si nos fijamos en los dos flujos, vemos que el predicho por el modelo numérico zigzaguea en torno a los bordes del mallado, mientras que el flujo del modelo PINN sigue una línea recta. Esto significa que el modelo PINN ha fallado, pues la predicción realizada, si bien menos caótica que la anterior, sigue siendo errónea, pues ha predicho otra clase de flujo (flujo laminar en este caso).

Visto esto podemos concluir que las simulaciones de fluidos están aún fuera del alcance de las PINNs. Quizá sí que den buenos resultados para flujos laminares, o para fluidos mucho más viscosos, ambos fenómenos representados por funciones mucho más suaves. Las funciones que rigen este sistema presentan variaciones muy fuertes en las regiones que contienen turbulencias, que es posiblemente una de las razones por las que la PINN ha tenido tantos problemas en este caso.

Capítulo 5

Conclusiones y Trabajos Futuros

En este capítulo se hará una revisión de los resultados obtenidos en la experimentación. En primer lugar se explicará qué inconvenientes y qué ventajas se han observado en las PINNs respecto del resto de métodos expuestos. Después se propondrán distintas vías de desarrollo para las PINN con el objetivo de solucionar estos inconvenientes, y de aumentar su facilidad de uso y versatilidad.

5.1. Conclusiones.

Como hemos observado durante el análisis de la experimentación, las PINNs presentan varios problemas. En primer lugar, únicamente hemos encontrado 2 funciones de activación que dieran lugar a resultados aceptables: la tangente hiperbólica y Softplus. El resto de funciones de activación consiguen aproximaciones bastante peores, como por ejemplo ReLU, o ni siquiera consiguen obtener resultados aceptables, como las funciones que descartamos en el primer experimento: Softmax y sigmoide.

Por otro lado es necesario hacer un comentario sobre los tiempos de entrenamiento necesarios para obtener buenos resultados. Durante la experimentación, la mayoría de las redes que´han obtenido mejores valores en las métricas utilizadas han necesitado gran cantidad de tiempo para ajustarse. Esto supone una gran desventaja respecto del estado actual de otros métodos disponibles hoy en día, pues el tiempo de computación necesario para obtener resultados con estos otros métodos es mucho menor que el tiempo de entrenamiento de las PINNs. En su estado actual, las PINNs reducen en gran medida el tiempo necesario para generar un modelo por lo simple que es el método de implementación. No

obstante, el tiempo de entrenamiento para un problema complejo, como el visto en la Sección 4.4, que necesite gran cantidad de épocas para ajustarse correctamente puede durar horas, mientras que un método numérico necesita unos minutos.

También es necesario recordar los errores que comete la PINN si el problema propuesto es demasiado complejo. Como vimos en la Sección 4.4, si la PINN recibe el mismo tipo de información que un método clásico, parece incapaz de realizar predicciones satisfactorias, aun empleando gran cantidad de épocas de entrenamiento. Por otro lado, si se incrementa la cantidad de información que ésta recibe, parece que las predicciones mejoran. Por tanto, si la cantidad de datos es insuficiente, y el problema a resolver es demasiado complejo, se puede esperar que las predicciones de la PINN sean bastante imprecisas.

Como hemos visto en los experimentos 2 y 3, las PINNs no responden bien a las discontinuidades o a datos no homogéneos. Esto quiere decir que si existe alguna clase de error de medición muy extendido, o en el experimento se pretende combinar condiciones iniciales contradictorias (como por ejemplo temperatura uniforme junto con un foco de calor), la PINN dará muchos problemas al ajustarse si no se introducen los datos con cuidado. Por esta razón es aconsejable revisar los datos disponibles y retirar cualquier anomalía de estos, y también sería una buena idea evitar discontinuidades muy fuertes en las muestras de datos.

Por otro lado, el método de construcción de modelos dado por las PINNs tiene varias ventajas. En primer lugar, el proceso de construcción de un modelo es mucho más simple, sobre todo si intervienen gran cantidad de variables en el problema, mientras que a la hora de construir un modelo basado en métodos numéricos suele ser necesario adaptarlo al problema, aplicando métodos más complejos. En definitiva, el incremento en la complejidad del problema no afecta mucho a la complejidad de la implementación de la PINN, mientras que sí que afecta en gran medida a la complejidad de otros métodos.

También hay que tener en cuenta que existen escenarios en los que ninguno de los métodos clásicos es suficiente. Los métodos que hemos utilizado hasta ahora asumen que se conoce el estado inicial del espacio a simular, y las condiciones de contorno del espacio durante todo el tiempo de experimentación, pero este no siempre es el caso. En aquellas ocasiones en que se disponga de suficientes datos de entrenamiento obtenidos mediante sensores, pero no se disponga de esa clase de información, las PINNs pueden suponer una buena alternativa, pues son capaces de obtener una predicción únicamente a partir de los datos de entrenamiento obtenidos de los sensores, y de las ecuaciones diferenciales propias del fenómeno que se esté simulando.

5.2. Trabajos futuros.

Como hemos visto en la sección anterior, las PINNs tienen varios inconvenientes: falta de funciones de activación aptas, tiempos de entrenamiento demasiado largos, resultados poco precisos y dificultades con las discontinuidades en los datos. Los tres primeros inconvenientes tienen una base común, que es la falta de funciones de activación pensadas para esta clase de problema. Como hemos visto durante la experimentación, en problemas complejos, las PINNs convergen muy lentamente a una solución fiable, en los casos en que la alcanzan. Esto se debe a que las funciones de activación que hemos utilizado no parecen responder muy bien a la técnica de entrenamiento de las PINNs, la cual tiene en cuenta las derivadas de distintos órdenes de dichas funciones. Encontrando funciones de activación que respondan bien a esta técnica, se podría reducir la cantidad de épocas de entrenamiento necesarias para obtener buenas predicciones, y quizá se obtendrían mejores resultados. Se proponen los siguientes pasos para la búsqueda de dichas funciones de activación:

- Realizar una revisión bibliográfica para encontrar funciones de activación menos extendidas, y realizar una implementación de estas.
- Proponer una serie de problemas variados que cubrieran varios fenómenos distintos (como el calor, el sonido y la mecánica de fluidos).
- evaluar el rendimiento de distintas PINNs utilizando las nuevas funciones de activación propuestas.

En cuanto a la mala respuesta de las PINNs a las discontinuidades en los datos, una posible solución sería implementar algún método de revisión de incoherencias en los datos. En el caso de SciANN, al construir la PINN es necesario introducir los datos como tuplas del tipo *[(índices,valores)]*. Por tanto, un módulo que revisara estas tuplas y encontrase incoherencias en los datos podría ser un buen comienzo.

Para terminar, vamos a hablar de otras modificaciones que se pueden añadir a las PINNs en su estado actual. Como ya hemos comentado, teóricamente es posible entrenar PINNs con diversas estructuras de red neuronal, tanto redes neuronales densas (DNNs) como redes neuronales convolucionales (CNNs). No obstante, las bibliotecas disponibles ofrecen únicamente la estructura de red neuronal densa (DNN). Sería interesante construir una implementación de PINNs que incluyese otro tipo de estructuras.

Por otro lado, se debería trabajar en las bibliotecas disponibles para construir mallados para PINN. Por ejemplo SciANN no contiene ningún módulo orientado a este aspecto, y la única herramienta disponible para construir mallados para este paquete es Numpy. Existen herramientas muy completas de construcción de mallados para el MEF, como por ejemplo GMSH [4]. Una biblioteca que permitiese generar mallados fácilmente, y que permitiese importar mallados construidos en GMSH fácilmente, simplificaría mucho el proceso de construcción de una PINN.

Bibliografía

[1] *Anaconda Software Distribution*. Ver. Vers. 2-2.4.0. 2020. URL: https://docs.anaconda.com/.

[2] L.C. Evans y American Mathematical Society. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, 2010. ISBN: 9781470411442. URL: https://bookstore.ams.org/gsm-19-r.

[3] Félix Fernández de la Mata. *Experimentación para el TFM del Máster en Ciencia de Datos e Ingeniería de Computadores de la UGR*. Sep. de 2021.

[4] Christophe Geuzaine y Jean-François Remacle. «Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities». En: *International Journal for Numerical Methods in Engineering* 79.11 (2009), págs. 1309-1331. DOI: https://doi.org/10.1002/nme.2579. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2579. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2579.

[5] Gisèle Ruiz Goldstein. «Derivation and physical interpretation of general boundary conditions». English. En: *Adv. Differ. Equ.* 11.4 (2006), págs. 457-480. ISSN: 1079-9389.

[6] Ehsan Haghighat y Ruben Juanes. «SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks». En: *Computer Methods in Applied Mechanics and Engineering* 373 (ene. de 2021), pág. 113552. ISSN: 0045-7825. DOI: 10.1016/j.cma.2020.113552. URL: http://dx.doi.org/10.1016/j.cma.2020.113552.

[7] Matthew Hancock. *18.303 Linear Partial Differential Equations, Fall 2004*. . Ene. de 2004. URL: https://www.researchgate.net/publication/37997455_18303_Linear_Partial_Differential_Equations_Fall_2004.

[8] Charles R. Harris y col. «Array programming with NumPy». En: *Nature* 585.7825 (sep. de 2020), págs. 357-362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[9] F. Hecht. «New development in FreeFem++». En: *J. Numer. Math.* 20.3-4 (2012), págs. 251-265. ISSN: 1570-2820. URL: https://freefem.org/.

[10] Oliver Hennigh y col. *NVIDIA SimNet^{TM}: an AI-accelerated multi-physics simulation framework*. 2020. arXiv: 2012.07938 [physics.flu-dyn].

[11] J. D. Hunter. «Matplotlib: A 2D graphics environment». En: *Computing in Science & Engineering* 9.3 (2007), págs. 90-95. DOI: 10.1109/MCSE.2007.55.

[12] Guofei Pang, Lu Lu y George Em Karniadakis. «fPINNs: Fractional Physics-Informed Neural Networks». En: *SIAM Journal on Scientific Computing* 41.4 (2019), A2603-A2626. DOI: 10.1137/18M1229845. eprint: https://doi.org/10.1137/18M1229845. URL: https://doi.org/10.1137/18M1229845.

[13] Tobias Pfaff y col. *Learning Mesh-Based Simulation with Graph Networks*. 2021. arXiv: 2010.03409 [cs.LG].

[14] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2019. URL: https://www.R-project.org/.

[15] M. Raissi, P. Perdikaris y G.E. Karniadakis. «Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations». En: *Journal of Computational Physics* 378 (2019), págs. 686-707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[16] Samuel H. Rudy y col. «Data-driven discovery of partial differential equations». En: *Science Advances* 3.4 (2017). DOI: 10.1126/sciadv.1602614. eprint: https://advances.sciencemag.org/content/3/4/e1602614.full.pdf. URL: https://advances.sciencemag.org/content/3/4/e1602614.

[17] Hadley Wickham y col. «Welcome to the tidyverse». En: *Journal of Open Source Software* 4.43 (2019), pág. 1686. DOI: 10.21105/joss.01686.